(NASA-CR-158633) OPERATING MANUAL FOR THE RRL 8 CHANNEL DATA LOGGER (Illinois Univ.) 124 p HC A06/MF A01 CSCL 09B

N79-24671

Unclas

G3/61 20895

OPERATING MANUAL FOR THE RRL 8 CHANNEL DATA LOGGER

by

E. J. Paluch
J. D. Shelton
C. S. Gardner

RRL Publication No. 502

Technical Report
May 1979

Supported by

Contract No. NASA NSG-5049

NATIONAL AERONAUTICS & SPACE ADMINISTRATION
Goddard Space Flight Center
Greenbelt, Maryland 20771

RADIO RESEARCH LABORATORY
DEPARTMENT OF ELECTRICAL ENGINEERING
COLLEGE OF ENGINEERING
UNIVERSITY OF ILLINOIS
URBANA, ILLINOIS 61801

OPERATING MANUAL FOR THE RRL 8 CHANNEL DATA LOGGER

by

E. J. Paluch
J. D. Shelton
C. S. Gardner

RRL Publication No. 502

Technical Report
May 1979

Supported by

Contract No. NASA NSG-5049

NATIONAL AERONAUTICS & SPACE ADMINISTRATION
Goddard Space Flight Center
Greenbelt, Maryland 20771

RADIO RESEARCH LABORATORY
DEPARTMENT OF ELECTRICAL ENGINEERING
COLLEGE OF ENGINEERING
UNIVERSITY OF ILLINOIS
URBANA, ILLINOIS 61801

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

This unit is a data collection device which takes measurements from external sensors at user specified time intervals. Three sensor ports are dedicated to temperature, air pressure, and dew point. Five general purpose sensor ports are also provided. Each sensor port supplies two sources of plus and minus 15 volts for sensor power. One of the power supplies is for sensors that stabilize quickly and the other power supply is for sensors that take longer than a few seconds to stabilize. (Sensor warm up time is under software control for maximum system flexability.)

After connecting the desired sensors the user specifies when measurements are to be recorded. The user enters the desired starting and stopping dates and times for the data collection. The warm up time for the "slow" sensors and the frequency of data collection must also be specified. If more than the three dedicated sensors are connected the additional ports must also be entered. This completes the initialization and the unit may be left running completely powered or it may be switched to a low power consumption mode.

While the system is running the user can switch full power on and interrogate the unit to examine any of the measurements that have already been recorded. The user can also examine the current readings on any of the sensors or the data that have been collected can be dumped to a peripheral device (a minicomputer, paper tape, etc.) if desired.

SECTION 1 OPERATION

1.1 FRONT PANEL DESCRIPTION

1. MAIN POWER SWITCH - This switch (1 in Figure 1) connects the power to the unit. The switch is a "lever locking switch" and must be pulled out before it can be operated. Since the data logger's memory is volatile, turning off the main power will destroy all stored information.

2. PROCESSOR POWER SWITCH - If the main power switch (1 on the front panel) is on, the processor power switch (2 on the Front Panel) will provide power to every part of the data logger except the RS-232 transmitting interface (I/O power). All sensors will be powered when this switch is on and the data logger will perform any user request if a terminal is connected and I/O power is on. (NOTE: If the data logger is supplying warm up power to the sensors and this switch is turned off the warm up power will also turn off and the next set of readings will be invalid. At any time, except when warm up power is on, this power may be switched off and the data logger will record all measurements correctly.)

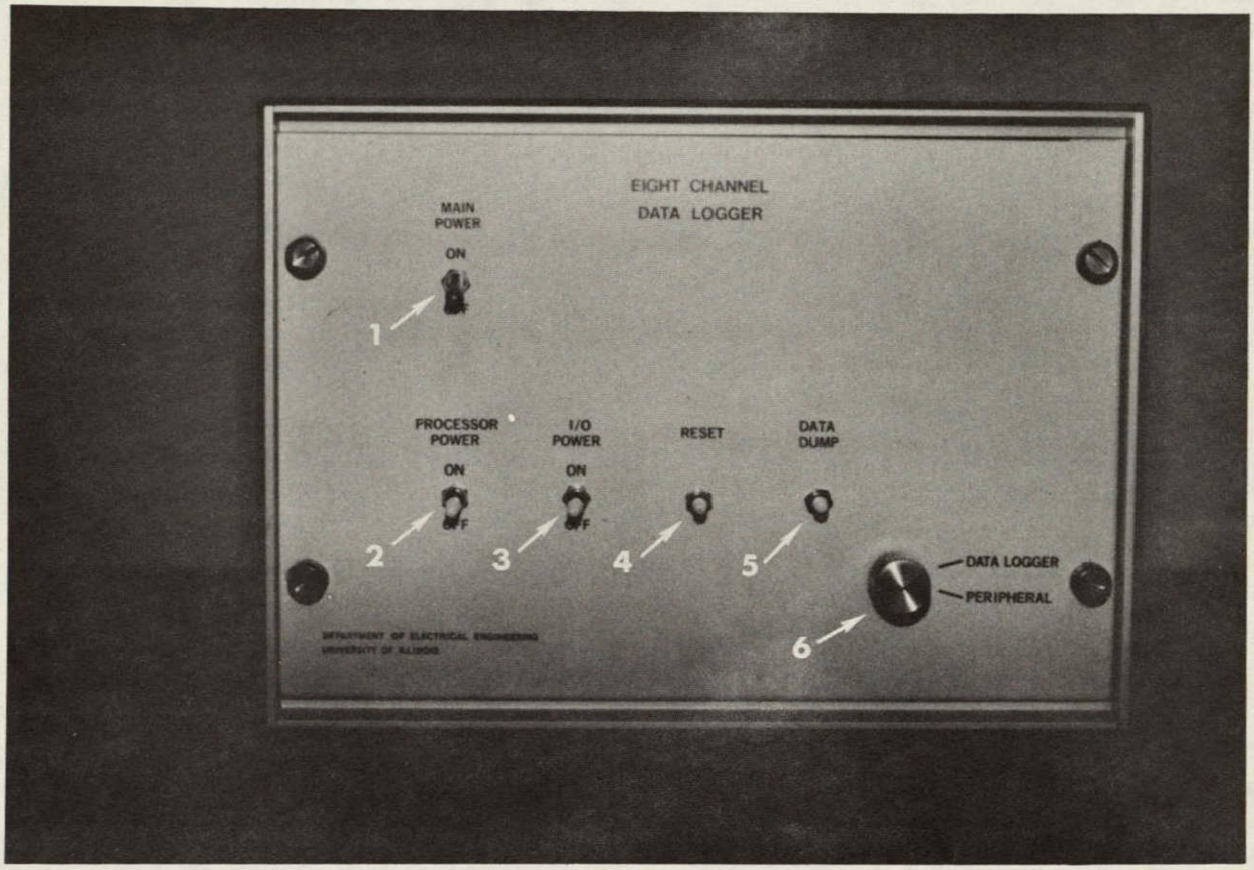


Figure 1

FRONT PANEL

3. I/O POWER SWITCH - This switch controls the power to the RS-232 transmitter interface to the terminal connection (8 in Figure 2) and to the peripheral connection (7 on the Back Panel). If this switch is off the terminal and peripheral can transmit to the data logger, but the data logger will not transmit to them. (NOTE: It is advisable not to transmit either from a terminal or a peripheral device to the data logger if I/O power is off and the processor power is on.) In most cases the I/O power and the processor power will be both on or both off.

4. RESET - This switch resets the processor and should be pressed after the user turns the data logger on. If the data logger stops responding to user inputs the reset can be used to clear the transmission lockup.

5. DATA DUMP - This switch is only active if the user has selected the data dump option from the user options page (see section 1.3-6, DATA DUMP OPTION). If active, depressing the switch will transmit all of the collected data to the terminal and peripheral connections.

6. DEVICE SELECTION SWITCH - If terminal is selected, two way communications occur between the terminal (teletype or video terminal) and the data logger. Most of the time the data logger will be used in this manner.

If the peripheral is selected, the terminal transmits to the data logger and receives data through the peripheral connector. The data logger transmits to the peripheral connector. This feature can be used to transfer data from the data logger to another computer and is described in section 1.3-6, DATA DUMP OPTION.

1.2 BACK PANEL DESCRIPTION

7. PERIPHERAL CONNECTION - This connection allows the data logger to communicate with a peripheral device (i.e., computer, minicomputer, papertape puncher, etc.) through a terminal. The data logger can also dump readings to the peripheral device. (The connections are listed in the appendix.)

8. TERMINAL CONNECTION - This connection is for a teletype or video terminal that will be used to communicate with the data logger. The terminal will enter parameters that will initialize the data logger and can also request information from the data logger.

9. POWER CONNECTION - This connector is for +12 volt power for the data logger.

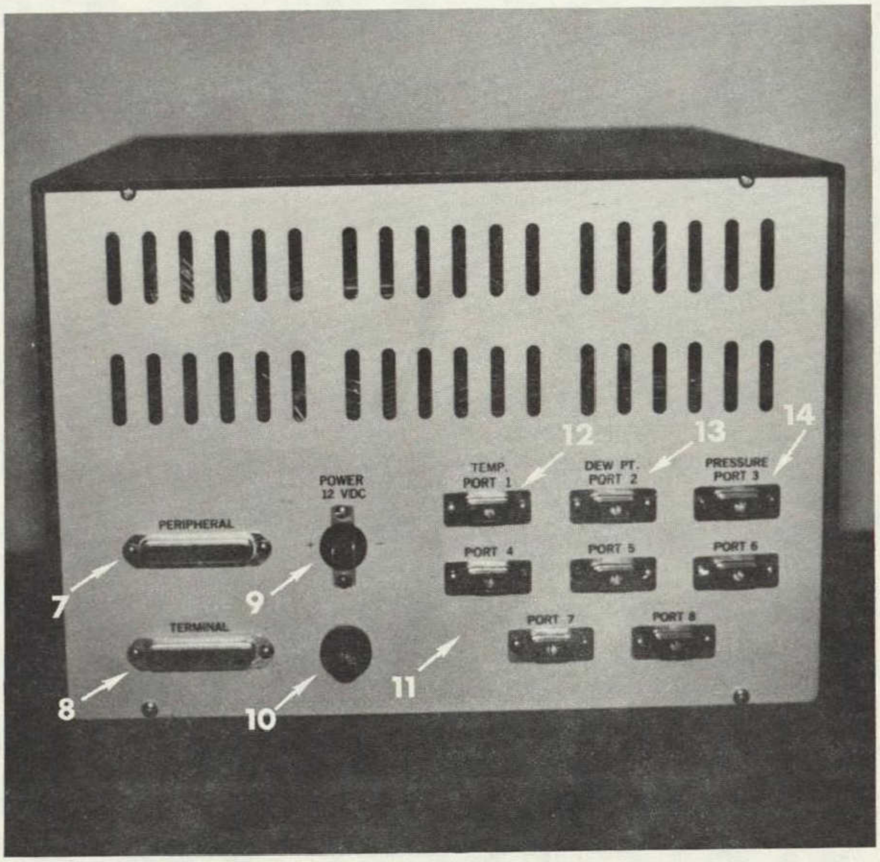


Figure 2

BACK PANEL

10. FUSE - A 1.5 Amp fuse should be used.

11. SENSOR PORTS - There are 8 sensor ports on the back panel. The ports are labeled 1-8 and the first three ports are dedicated. Each port connector has a coax signal connection and four power connections. (The connections for the ports are listed in the appendix.) The coax signal should be in the range of 0-5 Volts.

12. TEMPERATURE PORT - This port is dedicated to measuring temperature. 0-5 Volts corresponds to -30 - +70 degrees Centigrade.

13. DEW POINT PORT - This port is dedicated to measuring dew point. The 0-5 Volt range corresponds to -34 - +50 degrees Centigrade.

14. AIR PRESSURE PORT - This port is dedicated to measuring the air pressure. The 0-5 Volt range corresponds to 800 - 1100 millibars.

1.3 DESCRIPTION OF USER OPTION PAGE

The user has access to five basic routines that allow him to interact with the data logger. A list of these options is printed on the terminal whenever any key except CONTROL S, A, B, C, D or E is depressed and the system is not executing one of these routines. Samples of typical displays are included in each of the associated sections below. The underlined characters correspond to the user's responses.

1. LIST OF OPTIONS - The list of options is printed on the terminal whenever any key except CONTROL S, A, B, C, D or E is typed and the system is not executing one of these routines. It should be noted that the warm up power status is displayed prior to the option list. If warm up power is on, the PROCESSOR POWER switch should not be turned off immediately (this problem is discussed in section 1.1-2 of this manual). The following figure is an example of the user option page.

EXAMPLE: (user depresses the space bar and the system responds with:)

```
   WARM-UP POWER IS ON
   A) INITIALIZING PARAMETERS
   B) CURRENT SENSOR READINGS
   C) SEQUENTIAL READINGS
   D) INITIALIZATION
   E) DUMP
        (CONTROL 'S' STOPS PRINTOUT)
```

To invoke any option, the letter preceding it on the option page should be typed on the terminal. To exit any option prior to its end, CONTROL S may be typed.

2. OPTION A) INITIALIZING PARAMETERS - After the system has been initialized, the initializing parameters may be examined by typing A. As shown in the example contained in this section, the measurement starting date and time, ending date and time, warm up time in minutes and interval of measurements are displayed at the terminal.

EXAMPLE:

```
A

START      04/01      03:00
STOP       04/29      16:00
WARMUP TIME 02 MINS
FREQUENCY OF MEASUREMENTS     00:05
```

3. OPTION B) CURRENT SENSOR READINGS - Sensor outputs may be examined at any time with this routine. After typing B, a heading is printed which lists all ports to be interrogated. Temperature (port 1), dew point (port 2) and pressure (port 3) are dedicated ports and will always be printed. Any additional ports selected during system initialization will then be printed. To measure the sensor outputs, type R. Temperature and dew point readings have units of degrees centigrade while pressure readings have units of millibars. Any additional

sensors selected will yield readings with units of volts. As many sets of readings as desired may be accumulated. To exit this option, CONTROL S must be typed. This routine does not store these readings. Therefore, these readings cannot be retrieved at a later date by using the data dump option. If the data logger takes a reading during the execution of this option, the current line of sensor readings is stopped and the program waits for the entry of another R or CONTROL S. An example of this option is included below.

EXAMPLE:

```
B

TYPE 'R' FOR A READING, CONTROL 'S' WHEN DONE
  TEMP  DEW PT  PRESS  PORT1 PORT2 PORT3
R  6.72 - 3.81  983.78 1.853 1.812 3.062
R  6.72 - 3.81  983.78 1.848 1.812 3.062
```

4. OPTION C) SEQUENTIAL READINGS - This routine permits the user to examine data the data logger has accumulated. After typing C, the system requests a starting date and time followed by the number of readings that are desired for display. Dates are input in five-character fields by typing the number of the month in two digits, a slash (/) and the day of the month in two digits. The time is input in a five-character field by typing the two digit number of the hour, a colon (:) and the number of minutes in two digits. All two digit numbers must have unsurpressed leading zeros. The number of the hour is based on a 24 hour clock. A heading similar to the heading of option B

but including columns for date and time is printed. If any readings were taken on or after the specified date, they are printed as shown in the example in this section. When the requested number of readings has been printed or when the last reading taken has been displayed, the routine is exited. Similarly, if no readings were taken after the specified date and time, the processor exits the routine after printing the heading. As before, this routine may be terminated at any time by typing CONTROL S. If a reading (NMI) occurs during the data printout portion of this option, the current line of data is interrupted. The complete line is reprinted on the next line. NOTE: To avoid possible problems, care should be exercised to avoid entering data (dates, times and number of readings) when the data logger is scheduled to take a reading.

EXAMPLE:

```
C

START DATE?(MM/DD) 04/01      TIME?(HH:MM) 08:03
HOW MANY?(XXX) 003
 DATE  TIME   TEMP  DEW PT  PRESS  PORT1 PORT2 PORT3
04/01 08:05   1.25 - 7.09  989.12 1.565 1.607 3.152
04/01 08:10   1.25 - 7.09  989.12 1.563 1.604 3.152
04/01 08:15   1.25 - 7.09  989.12 1.563 1.604 3.152
```

5. OPTION D) INITIALIZATION - This option is used to prepare the data logger to take a series of readings. If data is stored in the system, it should be dumped before this routine is entered. After entering the initialization procedure, any old data stored in the system is no longer accessible. To

protect data in case this option is inadvertently entered, after D is typed the user is prompted with INITIALIZATION?(YES,NO). If YES is typed, this routine is entered. Any other character string will cause the processor to back out of this option.

If YES is entered, the user is asked for the current date and time. Format for the date and time are the same as that used in the sequential reading option. When the last character of the time is entered, an internal clock is reset and starts keeping track of elapsed time. The desired starting date and time are requested followed by a request for the date and time of the final reading to be taken Each date and time are entered in the same format as described in section 4. After the corresponding prompt, the number of minutes of warm up time desired should be entered as a two digit number. When asked for the frequency of measurements, the user should enter a three digit number corresponding to the number of minutes between readings.

If a mistake is made during the preceding part of this option, the ESCAPE key may be used. The user is then prompted to reenter the necessary data. The ESCAPE key may be used several times in order to step backwards through this routine and correct mistakes at any point in the option.

The second part of this routine is now entered. The user is asked if he desires any additional ports. If NO is typed, only the first three dedicated ports are used for temperature, dew point and pressure measurements. If YES is typed, the user is asked which port he wishes to use. He responds by typing a number between 1 and 8 corresponding to the desired port. He is then asked once more if he desires any additional ports. This process is repeated until all the desired ports are specified and a NO is entered. At this point the user is prompted REASSIGN PORTS?(YES,NO). It is suggested that the port assignments be checked and if an error is found, type YES. This causes the port assignment portion of this option to be reexecuted. If NO is typed, the processor exits this option. It should be noted that an error will result if the same additional port is specified more than once in the second part of this option (i.e., if port 2 is selected twice as an additional port). However, ports 1, 2 and 3 may be specified as additional ports even though they are dedicated for temperature, dew point and pressure measurements. This may be desireable because as dedicated ports, the voltages present at ports 1 and 2 are stored as 8 bit binary words while if they are selected as additional ports, the voltages are stored as 12 bit binary words the second time they are measured. This permits the voltages present at ports 1 and 2 to be stored with greater precision. The remaining port voltages are always stored as 12 bit binary words.

EXAMPLE:

```
D

INITIALIZATION?(YES,NO)
YES
      DATE?(MM/DD) 03/31     TIME?(HH:MM) 17:22
START DATE?(MM/DD) 04/01     TIME?(HH:MM) 03:00
STOP  DATE?(MM/DD) 04/29     TIME?(HH:MM) 16:00
WARMUP TIME?(MM) 02
FREQUENCY OF MEASUREMENTS?(XXX) 005
ADDITIONAL SENSORS?(YES,NO)
YES
PORT NUMBER?(1-8)1
ADDITIONAL SENSORS?(YES,NO)
YES
PORT NUMBER?(1-8)2
ADDITIONAL SENSORS?(YES,NO)
YES
PORT NUMBER?(1-8)3
ADDITIONAL SENSORS?(YES,NO)
NO

REASSIGN PORTS?(YES,NO)
NO
```

NOTE: No provision has been made to keep track of the year in which a measurement is made. Because of this, if the system is initialized in the year before readings are to begin, the months of the year in which readings occur are numbered, 13, 14, etc. Further, the data logger treats all months in the second year as if they contain 28 days. Since this sytem was not intended to run for periods of more then one month unattended, this should present no problem. As an example, if the data logger is initialized on December 30, 1979 and is to start taking readings on January 3, 1980, the starting date used in the initialization routine would be 13/03. (Regardless of the year the data logger assumes February has 28 days.)

6. OPTION E) DATA DUMP - This option is designed to permit the data logger to be interfaced to an external device through the peripheral connector in order to dump the collected readings. The routine is entered by typing E. At this point, any key may be typed except CONTROL S and it will merely be echoed to the terminal or peripheral (depending on the position of the DEVICE SELECTION switch as described in section 1.1-6). To dump data to an external device (through the peripheral connector), place the DEVICE SELECTION switch to PERIPHERAL. The external device now transmits to the terminal and the terminal transmits to the external device through the data logger. If the data dump switch is depressed, the data logger transmits a short header followed by all the data it has collected. When all of the data has been transmitted, the data logger once again echos all input characters. Control S must be used to leave this routine The formats of the header and data are now discussed.

a). HEADER - After the data dump switch is depressed, the first six printed lines contain the initializing parameters. All lines in the header are left justified and separated by a carraige return, line feed and two delete characters. The first two lines consist of starting date and time and ending date and time respectively. All dates and times are transmitted in five character fields identical to the format required to enter dates and times in the initialization option. Separating each date and time is a field of five blanks. Line three contains the warm up time represented as a two digit decimal number. Line four contains the frequency of measurements in the same five-character field as described for the starting and stopping times. A two digit hexadecimal number in the fifth line gives the number of data words stored each time a reading is taken. Another two digit hexadecimal number in the sixth line tells which ports are selected for measurement in addition to the three dedicated ports (which are always measured). This information is conveyed by examining the binary equivalent of the number. Each of the eight binary digits corresponds to a port. If a digit is one, the corresponding port is selected for additional measurement. Letting the least significant digit be bit 0 and the most significant digit be bit 7, the correspondences of binary bits in this eight bit word and ports are as follows;

BIT 0	-	PORT 1
BIT 1	-	PORT 2
BIT 2	-	PORT 3
BIT 3	-	PORT 4
BIT 4	-	PORT 5
BIT 5	-	PORT 6
BIT 6	-	PORT 7
BIT 7	-	PORT 8

b) DATA FORMAT - The seventh and all following lines are devoted to collected data. Each line corresponds to a separate reading and is set off from succeeding lines by a carraige return, line feed and two delete characters as before. Within each line the unsigned hexadecimal data words are separated from each other by a single space. The first two data words of each line are two digits long. They represent the temperature port voltage (port 1) and the dew point port voltage (port 2) respectively. The remaining data words on each line are three digits long. The first of these numbers always represents the pressure port voltage (port 3). From this point on, the data words represent voltages on additional ports selected by the user starting with the smallest port number and continuing to the largest port number. To convert these hexadecimal data words to corresponding voltages, the decimal equivalent of each data word must first be obtained. Equation 1 is used to convert the first two data words. The remaining data words are converted using equation 2.

1) DATA WORD X 5/256 = V(PORT)

2) DATA WORD X 5/4096 = V(PORT)

In order to convert port voltages on ports 1, 2 and 3 to temperature, dew point and pressure, the conversions listed below may be used;

PORT 1	TEMP(DEGREES C) = V(PORT1) X 20.00 - 30.00
PORT 2	DEW PT.(DEGREES C) = V(PORT2) X 16.80 - 34.00
PORT 3	PRESSURE(mb) = V(PORT3) X 60.00 + 800.00

1.4 HARDWARE SETUP

Before applying power the system must be configured correctly. This involves checking the memory configuration, the power switching board, and the baud rate.

The memory must start at address 4000 (hex) and be contiguous upward. Each memory board has a dip switch which determines the location of the memory board. The starting addresses of the memory are written next to each switch on the board. (Either the full four digits or the first two digits of the address are given. For example, on the 2K memory board 42 indicates the starting address is 4200 (hex).) The memory addresses must not overlap, therefore, if both 2K and 4K memory boards are used in the data logger the 4K boards must reside at lower addresses than the 2K boards. When a 4K board and a 2K board are used, the 4K board must be set at 4000 and the 2K board at 5000 (50). (In hex 1K=400, 2K=800, 3K=C00, 4K=1000.) It is important that only one switch is closed at any time.

The available memory must be checked to insure that enough is available to record the number of readings desired. The following two formulas may be used to calculate how much memory is required or how many readings may be taken:

ADDSEN=The number of additional ports selected.
MEM=The amount of memory in system.
READINGS=The total number of readings you want to take.

Memory needed(bytes)=$\text{READINGS}\times\lceil 2+1.5\times(\text{ADDSEN}+1)\rceil+64$

Number of readings allowable=$(\text{MEM}-64)/\lceil 2+1.5\times(\text{ADDSEN}+1)\rceil$

The symbols $\lceil,\rceil$ indicate "smallest integer larger than".

EXAMPLES:

For example, if one additional port is selected and readings are taken every hour for 31 days, then the amount of memory needed would be

$$\text{MEMORY}=(31\times 24)\times\lceil 2+1.5\times(1+1)\rceil+64$$

$$=744\times\lceil 5\ +64\rceil$$

$$=3784$$

$$\text{APPROX.}=3.8\text{K}$$

If no additional ports are selected how long can readings be taken every half hour with 6K of memory?

$$\text{NUMBER OF ALLOWABLE READINGS}=(6\text{K}-64)/\lceil 2+1.5*(0+1)\rceil$$

$$=6080/\lceil 3.5\rceil$$

$$=1520$$

$$\text{Number of Days}=\ 31.5$$

It is important to remember that the data logger can not tell if it tries to store readings and has exceeded the available memory. If this happens, sequential readings and data dump will print out incorrect values for the readings that it tried to store after exceeding the available memory. The data logger may be configured with up to 12K bytes of memory.

A dip switch for the baud rate selection is located on the processor board. All communication between the data logger, terminal, and peripheral must occur at the same rate. The available baud rates are 110, 150, 200, 300, 600, 1200, 1800, and 2400. It is important that only one of the switches is closed at any one time!

The power board has three dip switches that control the power switching relays. If the processor power is going to be left on during the entire time the data logger is operating then these three switches should be left off. This will disable the relays and extend their life. If the data logger will operate with the power off, these three switches must be on.

1.5 RECOMMENDED STARTUP PROCEDURE

1. Configure the memory correctly.

2. Ensure that sufficient memory is available.

3. Check the relay dip switches on the power board for the correct position.

4. Check the baud rate.

5. Turn off the MAIN POWER switch.

6. Connect the +12VDC to the Back Panel.

7. Turn on the PROCESSOR POWER and I/O POWER switches.

8. Turn the DEVICE SELECTION switch to data logger.

9. Connect the sensors.

10. Turn on the MAIN POWER switch.

11. Depress the RESET switch.

12. Use the software options as desired. When initializing the data logger remember;

    1. The internal clock is reset when the last digit of the present time is entered.

    2. The starting date/time must be after the present date/time.

    3. The stopping date/time must be after the starting date/time.

    4. The warm up time must be less than (stopping date/time - present date/time).

    5. The frequency of measurements must be greater than the warm up time. If the warm up time is greater than the desired frequency of measurements, then set warm up time

equal to zero and always leave the processor power on.

6. If a correction is made when specifying the additional ports all ports must be respecified.

If any of these rules are violated the data logger will not work correctly.

13. Use current readings to check sensor connections.

14. Turn off the I/O power when done.

15. If the processor power does not have to be left on, turn it off. Turn off processor power only when informed by the user option page that warm up power is off.

Once initialized the system can be turned on again as follows:

1) Turn I/O power on.

2) Turn the processor power on.

3) Depress the reset switch.

The terminal and/or peripheral can be connected (or disconnected) to the data logger at any time.

SECTION 2 HARDWARE DESCRIPTION

2.1 SYSTEM DESCRIPTION

This data logger was designed using the Motorola 6800 microprocessor family. A detailed understanding of the system operation requires familiarity with this 8-bit microprocessor family. However, the general overview which follows requires only a broad knowledge of a few of the 6800 family components.

a). The microprocessor (6802) has an eight bit bi-directional data bus and a sixteen bit address bus. Included on the chip are 128 bytes of random access memory (RAM) which are used as scratchpad memory and stack. Three other inputs of interest are IRQ, NMI and RESET. When IRQ or NMI is driven to zero volts, an interrupt or non-maskable interrupt, respectively, is initiated. The processor then starts executing routines to service these interrupts before returning to its original activity. When RESET is driven low, processor activity halts and its registers are initialized. When RESET goes high, the processor starts executing the restart routine.

b). The ACIA (M6850) is an Asynchronous Communications Interface Adapter. This device interfaces the microprocessor to a serial communications line. It requires an external clock to establish the baud rate.

c). The PIA (M6821) is a Peripheral Interface Adapter. This chip contains two eight bit ports and additional inputs which may be used to trigger interrupt requests to the processor. Each port (referred to as ports A and B) may have lines individually selected as inputs or outputs by software. In addition, port A is CMOS compatible.

The data logger is broken into the following five assemblies (see Figure 3).

1). PROCESSOR

2). CLOCK

3). A/D CONVERTER

4). MEMORY

5). POWER SWITCHING

Each assembly is housed on a separate printed circuit board. The memory, however, can be expanded simply by adding cards until a maximum of 12K bytes of RAM are used. This permits more data storage and thus longer periods of operation before dumping data is required.

The following sections briefly describe each board and indicate how boards interact with each other. A voltage followed by (s) is a switched voltage and is present only when the processor is powered. When +15 and -15 volts are referred to and no (s) follows them, the voltages are present when warm up power is on. If +5 volts is not followed by (s), it is the

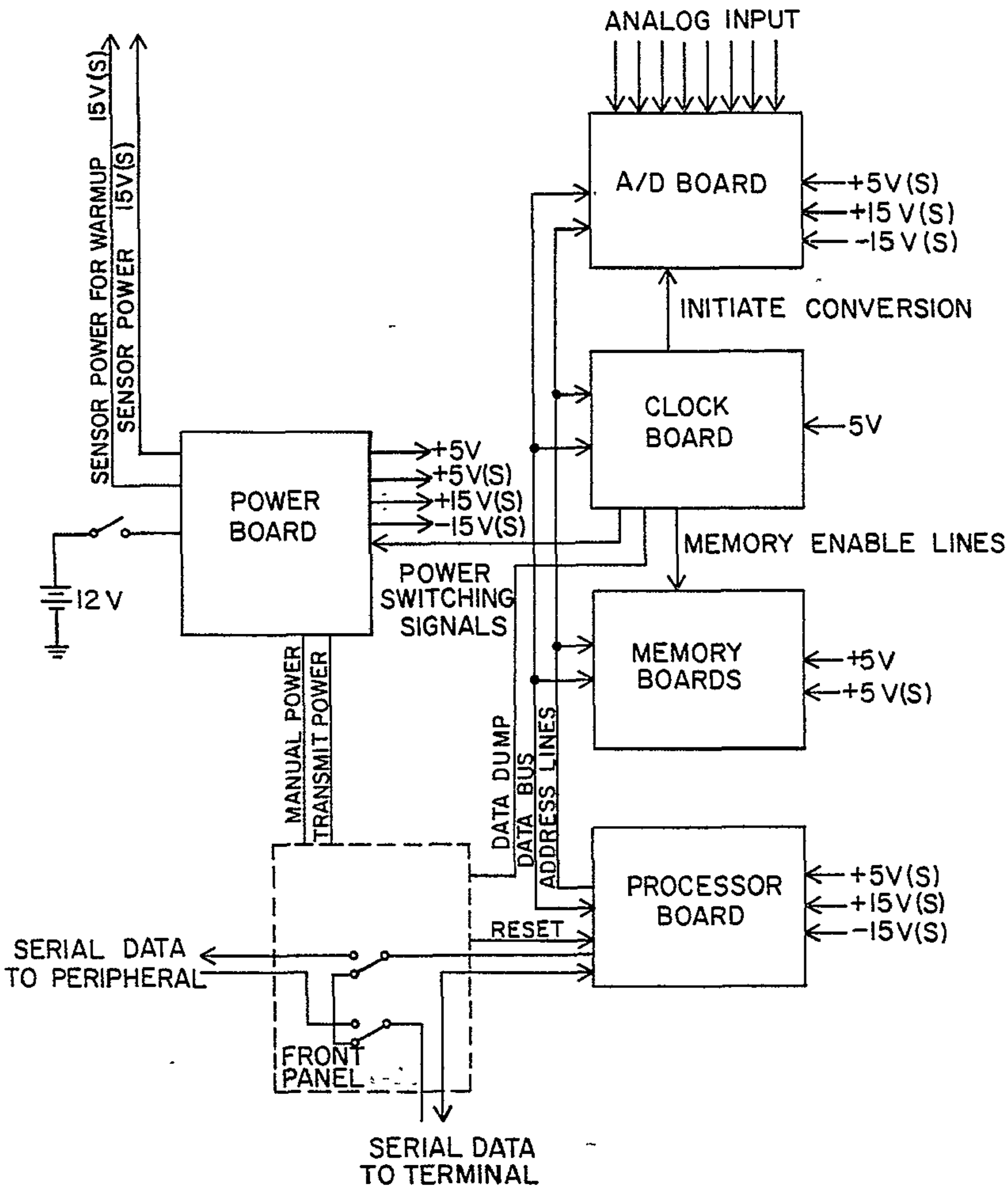


FIGURE 3

DATA LOGGER BLOCK DIAGRAM

unswitched voltage which is present whenever the MAIN POWER switch is on and +12 volts are applied to the system.

2.2 PROCESSOR BOARD

The processor board controls the overall operation of the system and provides serial communications with external devices via an RS-232C interface over which serial ASCII characters with even parity are transmitted and received. A block diagram of the board (as seen in Figure 4) is utilized to permit a general discussion of the board's operation. Detailed schematics of all the boards may be found in the appendix.

a). Power source requirements are listed below.

VOLTAGE	CURRENT (TYPICAL)	CURRENT (MAXIMUM)
+5(s)	400 ma	700 ma
+15 and -15	14.6 ma	19.3 ma

b). Operation - The processor may be broken into four basic sections.

Processor Section - The bulk of the processor board is devoted to the processor section. It is organized around the M6802 microprocessor. Decoding of high order address lines to enable the clock PIA, A/D PIA, ACIA and EPROMS (eraseable programable read only memories) is achieved with a 4 line to 16 line decoder (74154). Reset signals for PIA and processor initialization are generated here through a combination of RC

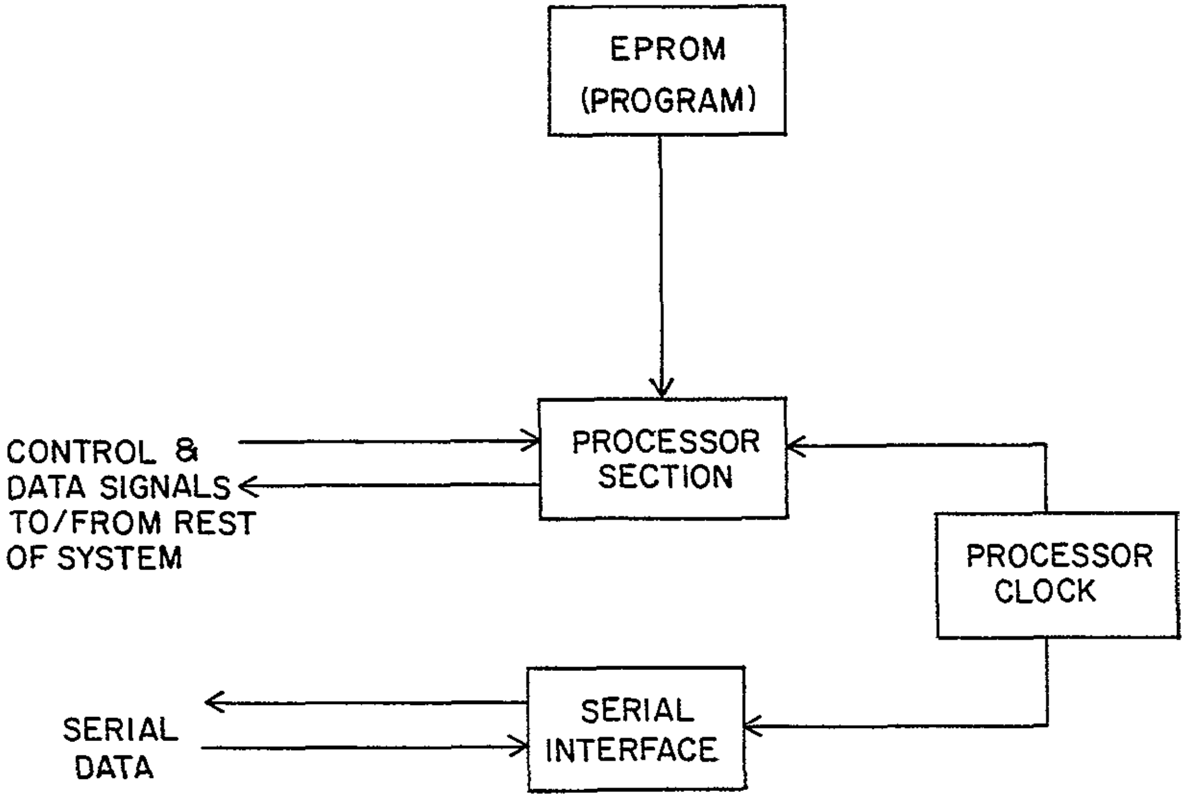


FIGURE 4

PROCESSOR BOARD BLOCK DIAGRAM

networks and schmidt triggers (7414). In order to interface the processor to the (relatively) slow EPROMs, the memory ready signal (MR) is held low for 870 nanoseconds after VMA (valid memory address) and E (enable) go high, thus slowing down the processor when it accesses external devices. Finally, read/write signals ($R/\overline{W} + MR$) and enable ($E \cdot VMA$) for use by memory boards originate in this section.

Processor Clock - Basic timing for both the processor and the serial interface portions of the processor board are provided by the processor clock. It consists of a baud-rate generator (MC14411) and a 1.843 MHz crystal. A buffered 1.843 MHz signal drives the processor while eight signals ranging from 1.7588 KHz to 38.4 KHz drive the serial interface. This variety of frequencies permits manual selection of baud rates from 110 to 2400 baud.

Serial Interface - This section transmits and receives asychronous serial data via the ACIA and makes it available to the processor on the eight bit data bus. Upon receipt of a complete, even parity ASCII word, an ACIA generated interrupt request causes the processor to execute the IRQ routine. The TTL transmit and receive lines from the ACIA are interfaced to RS-232C levels through standard interface adapters (M1488 and M1489 chips).

EPROM - The program for the data logger is contained in two

INTEL 2716 EPROMS. This yields a total program space of 4K bytes.

2.3 CLOCK BOARD

The clock board times the intervals between periods of processor activity such as data collection and switching on warm up power. It is also the source of various control signals used by other boards in the data logger.

a). Power supply requirements are listed below.

VOLTAGE	CURRENT (TYPICAL)	CURRENT (MAXIMUM)
+5	1 ma	9 ma
+5(s)	120 ma	300 ma

b). Operation - Timing is achieved by a combination of crystal controlled oscillator, counters, shift registers and coincidence gates. The crystal accuracy is approximately 0.01% over the temperature range -25 to +85 degrees centigrade. A one cycle per minute pulse is obtained from a 262.144 KHz oscillator and counting circuits. This signal increments a 12 bit counter. A comparison occurs between the contents of this counter and a 12 bit shift register containing the number of minutes in the desired interval. As soon as the two numbers match, a monostable multivibrator is triggered. This signal resets the minute counter, triggers an NMI request to the processor (in conjunction with the clock PIA) and exits the board to turn on all power at the power board.

The monostables which turn power off are also located on this board. In order to conserve power, the PIA which controls them is powered down except when the processor is in operation. To prevent false triggering, two bits of different levels are required from the PIA before either monostable is triggered. The same technique is used to reset the clock when the system is first initialized and to disable RAM whenever the processor is not powered.

Two more lines from the clock PIA are used elsewhere in the data logger. One triggers a conversion cycle by the A/D converter. The remaining line is configured as an input and responds to the data dump switch on the front panel. Table 1 lists the PIA port lines and their uses.

2.4 A/D BOARD

This board interfaces 8 single ended analog inputs to the processor. The digitized data is read by the processor as a 12 bit CSB (complementary straight binary) word.

a). Power supply specifications are listed below.

TABLE 1

Clock PIA

PORT A

BIT	PIN	FUNCTION
Ø	2	Reset Time
1	3	Clock for Shift Registers
2	4	Data for Shift Registers
3	5	Enables Power Down Monostables
4	6	Power Off (2)
5	7	Power Off (3)
6	8	Memory Enable
7	9	Memory Enable

PORT B

BIT	PIN	FUNCTION
2	12	Turns Off Interrupt
3	13	Data Dump Switch
5	15	Initiate A/D Conversion

A/D PIA

PORT A

BIT	PIN	FUNCTION
Ø	2	DØ (LSB)
1	3	D1
2	4	D2
3	5	D3
4	6	D4
5	7	D5
6	8	D6
7	9	D7

PORT B

BIT	PIN	FUNCTION
Ø	1Ø	D8
1	11	D9
2	12	D1Ø
3	13	D11 (MSB)
4	14	Conversion Status
5	15	Channel Select (SØ)
6	16	Channel Select (S1)
7	17	Channel Select (S2)

VOLTAGE	CURRENT (TYPICAL)	CURRENT (MAXIMUM)
+5(s)	150 ma	278 ma
+15 and -15	21.3 ma	23.0 ma

b). Operation - Three PIA lines are used to select the desired analog input at a CMOS switch (14529B). Before reaching the A/D converter (ADC80AG-12), the signal is buffered by a unity gain non-inverting operational amplifier (OP-02). The DC offset of this op-amp is adjusted by a potentiometer located on this board.

A conversion is initiated by (INIT CONV) • (CONV STATUS) where INIT CONV is the processor controlled signal from the clock board PIA and CONV STATUS is a status bit from the A/D converter. A low on CONV STATUS indicates that no conversion is currently occurring. Upon satisfaction of these requirements, a monostable is triggered which ultimately yields a 2.0 microsecond pulse to the A/D converter. This triggers a conversion cycle which takes 25 microseconds to complete. Upon completion, the status bit goes low informing the processor, through the A/D PIA, of the conversion status and enabling subsequent conversions. Table 1 lists the A/D PIA port lines and their uses.

2.5 MEMORY BOARDS

Two types of CMOS memory boards are available. Both types are similar except in memory capacity (2K bytes versus 4K bytes) and in the differences in address decoding this necessitates.

a). Power requirements are listed below. Standby currents pertain to the boards' current requirements when the boards are not being accessed by the microprocessor. Operation current refers to the current requirements when the board is being written to or read from by the microprocessor.

CURRENT	4K BOARD		2K BOARD	
VOLTAGE	TYPICAL	MAXIMUM	TYPICAL	MAXMUM
+5 (STANDBY)	8	5 ma	8	80
+5 (OPERATION)	40 ma	50 ma	80 ma	96 ma

(UNLESS NOTED, ALL CURRENTS ARE IN MICROAMPS)

b). 4K Byte Board - Eight 2K X 4 bit Harris 6514 CMOS memory chips form the nucleus of this board. Address decoding is performed in such a manner as to permit each board to be located at hex locations 4000, 5000, 6000 or 7000 in the memory space. These locations are selected by a DIP switch mounted on the board. The board may be enabled or disabled under processor control by MEM EN and $\overline{\text{MEM EN}}$. This combination of two signals to enable memory prevents erroneous data being written into the memory boards when processor and PIAs are powered down.

c). 2K Byte Board - Sixteen 1K X 1 bit Harris 6508 CMOS memory chips are utilized on this board. Operation of this board is similar to the 4K board except that the board may be located at 4000, 4800, 5000, 5800, 6000, 6800, 7000 or 7800 in the memory space.

2.6 POWER SWITCHING BOARD

The power switching board generates all the required supply voltages and switches +5, +15 and -15 volt supply lines in order to conserve power.

a). Power requirements are listed below.

		QUIESCENT CURRENT	
VOLTAGE	DEVICE	TYP	MAX
+12	LM 240LH	1 ma	5 ma
	LM 209K	2 ma	8 ma
	A12/D15/150/Z	EFFICIENCY RATED AT 70% AT FULL LOAD (150 ma AT +15 AND -15 VOLTS)	

b). Operation - Two five volt regulators and one plus and minus fifteen volt converter are utilized on this board. The LM240LH +5 volt regulator provides for the low current requirements of the memory boards and clock board and is therefore always in operation. The higher current requirements of the A/D and processor boards are handled by the LM209K. In order to conserve power, the input to this regulator is switched off when the processor and A/D boards are not in use. The output of the A12/D15/150/Z plus and minus fifteen volt voltage converter is used to drive sensors (external to the data logger), components on the A/D board and the RS-232C transmitter on the processor board. The input and outputs to this device are switched in such a manner as to permit a warm up period for sensors during which the processor and A/D boards are not powered.

Voltage switching is accomplished with three Teledyne 720-5 magnetic latching relays. These relays control inputs and outputs of the regulators and converter on this board. Signals controlling these relays originate on the clock board and have the following effects;

POWER ON (1) Turns on all switched voltages

POWER OFF (2) Turns off all switched voltages

POWER OFF (3) Turns off switched voltages except warm up voltages

If switched operation is not desired (i.e. when power need not be conserved) switches have been provided to disconnect these signals and thus extend relay life.

2.7 SYSTEM POWER REQUIREMENTS

As previously mentioned, only 12 vdc is required to power the data logger and all external sensors. The current requirement of the system is dependent upon the activity of the system.

This may be broken into three areas;

Period 1) All voltages on (processor taking readings).

Period 2) Warm up power on, all switched voltages off (sensors warming up).

Period 3) Warm up power and all switched voltages off (waiting for next NMI).

The approximate current requirement in milliamps for each of these periods is given below;

	TYPICAL	MAXIMUM
Period 1)	(820 + R/0.65)	(1300 + R/0.65)
Period 2)	(10 + RW/0.65)	(40 + RW/0.65)
Period 3)	3	40

In the above, R is the total current required by all sensors and RW is the current required by sensors powered by the warm up voltages. Although the length of Period 1 is dependent upon the number of sensors selected during the initialization procedure, it is typically on the order of 0.75 seconds for three sensors.

If the system is to be powered by batteries, the required amp hour rating may be found by multiplying the current drain associated with each of the above periods by the total time each of these periods occurs in the anticipated period of system operation.

SECTION 3 SOFTWARE DESCRIPTION

The software controls all of the hardware in the data logger, the selection of which power is on or off and the acquisition of data. A detailed description of the software (the variables used and the function of each subroutine) is given in the program listing in the appendix. (The reader is advised to read this section first before trying to understand the program listing.)

Whenever a user turns the system on and presses reset, or anytime the clock board finishes timing an interval and turns on the power the microprocessor starts the RESET routine (see figure 5). The reset routine can be divided into two parts; initialization of the newly powered system, and the user options. The reset routine initializes the stack pointer in order that stack operations and subroutine calls can be done. The PIAS are then initialized. Since turning on the power may change the timing interval in the clock, the clock is momentarily disabled and the time for the NMI is reset. Some of the on board RAM variables are cleared (specifically WHERE and BACKUP).

The second part of the reset routine waits for the user to type a key. If the key corresponds to a user option, the option is entered and the desired information is displayed or the user is prompted further. If the user types an invalid key the user options are retyped. NOTICE: When the user is at the user option page the memory is always disabled; at NO other time should the processor power be turned off.

The hardware interrupt (IRQ) is connected to the ACIA. Thus, whenever a character is received an IRQ interrupt is triggered. The IRQ flowchart is shown in figure 6. The received character is first checked for validity (even parity is used). An invalid character results in the printing of a question mark. A valid character is printed and stored in the location RECEIVE.

If the received character is an ESCAPE then BACKUP is checked. If BACKUP is zero the escape does nothing and a RTI (Return from Interrupt) is executed. If the escape is valid, the correct number of words are removed from the stack (until address STRIP is reached), a '*DEL*' is printed, and the user jumps back an input line. If the received character is a CONTROL S, several words from the stack are removed (as if a RTI was executed) and the user is returned to the options page. Every time a valid character is received a bit in location SIRQ is set. This tells the user option page that the user has pressed a key and would like something done for him.

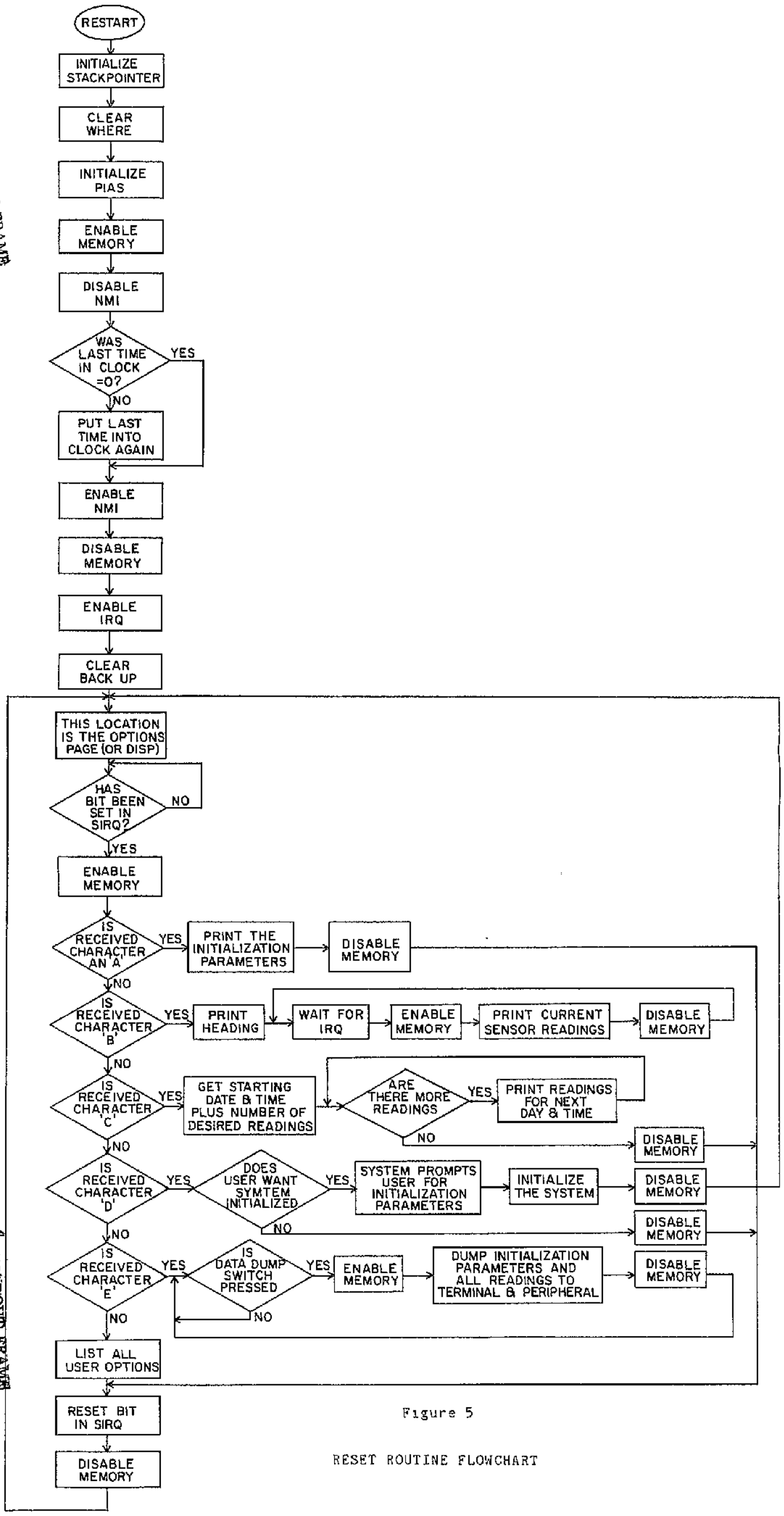

RESTART
INITIALIZE STACKPOINTER
CLEAR WHERE
INITIALIZE PIAS
ENABLE MEMORY
DISABLE NMI
WAS LAST TIME IN CLOCK =0?
YES
NO
PUT LAST TIME INTO CLOCK AGAIN
ENABLE NMI
DISABLE MEMORY
ENABLE IRQ
CLEAR BACK UP
THIS LOCATION IS THE OPTIONS PAGE (OR DISP)
HAS BIT BEEN SET IN SIRQ?
NO
YES
ENABLE MEMORY
IS RECEIVED CHARACTER AN 'A'
YES
PRINT THE INITIALIZATION PARAMETERS
DISABLE MEMORY
NO
IS RECEIVED CHARACTER 'B'
YES
PRINT HEADING
WAIT FOR IRQ
ENABLE MEMORY
PRINT CURRENT SENSOR READINGS
DISABLE MEMORY
NO
IS RECEIVED CHARACTER 'C'
YES
GET STARTING DATE & TIME PLUS NUMBER OF DESIRED READINGS
ARE THERE MORE READINGS
YES
PRINT READINGS FOR NEXT DAY & TIME
NO
DISABLE MEMORY
NO
IS RECEIVED CHARACTER 'D'
YES
DOES USER WANT SYMTEM INITIALIZED
YES
SYSTEM PROMPTS USER FOR INITIALIZATION PARAMETERS
INITIALIZE THE SYSTEM
DISABLE MEMORY
NO
DISABLE MEMORY
NO
IS RECEIVED CHARACTER 'E'
YES
IS DATA DUMP SWITCH PRESSED
YES
ENABLE MEMORY
DUMP INITIALIZATION PARAMETERS AND ALL READINGS TO TERMINAL & PERIPHERAL
DISABLE MEMORY
NO
NO
LIST ALL USER OPTIONS
RESET BIT IN SIRQ
DISABLE MEMORY

Figure 5

RESET ROUTINE FLOWCHART

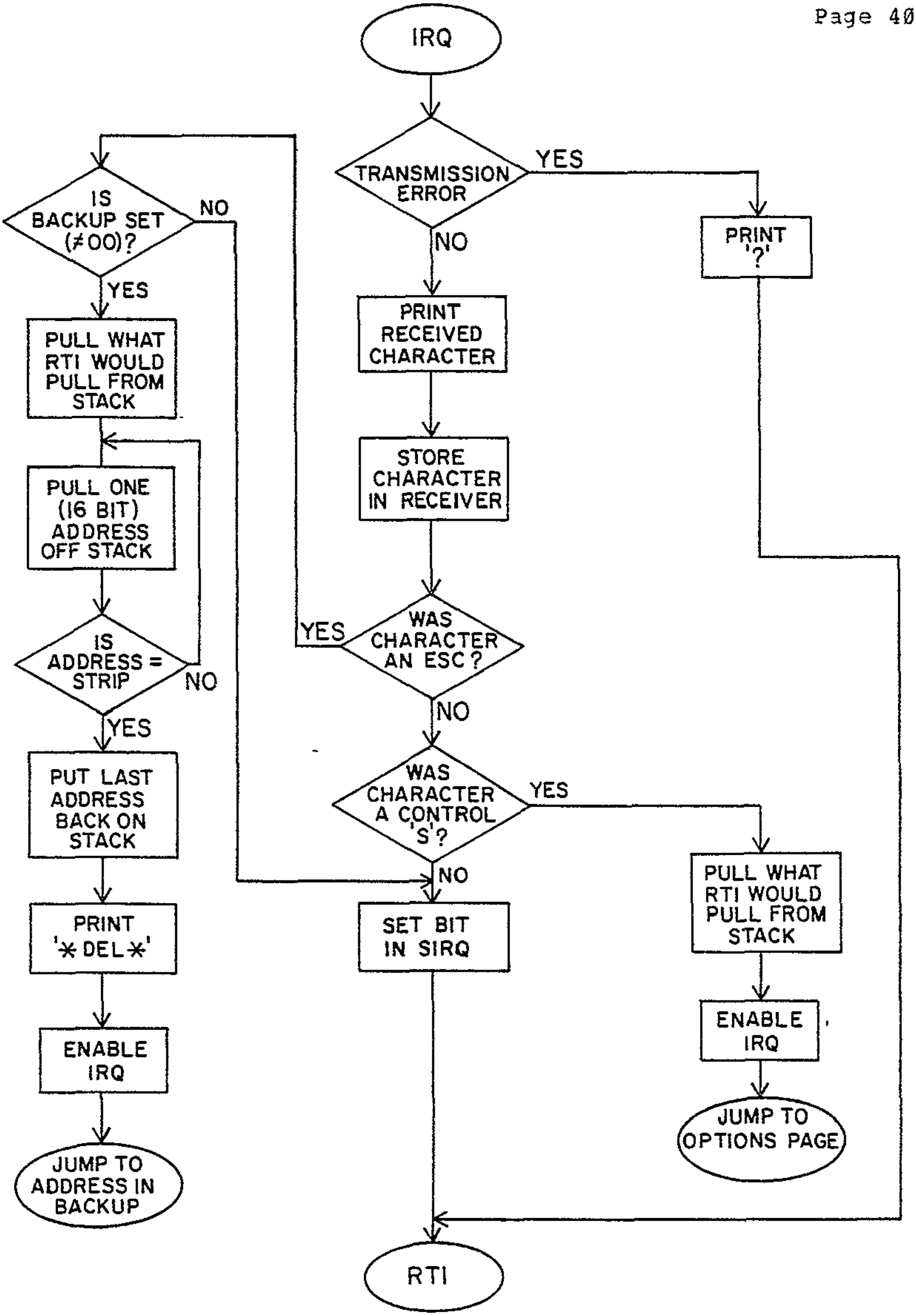

IRQ
TRANSMISSION ERROR
YES
PRINT '?'
NO
PRINT RECEIVED CHARACTER
STORE CHARACTER IN RECEIVER
WAS CHARACTER AN ESC ?
YES
NO
WAS CHARACTER A CONTROL 'S'?
YES
NO
SET BIT IN SIRQ
PULL WHAT RTI WOULD PULL FROM STACK
ENABLE IRQ
JUMP TO OPTIONS PAGE
RTI
IS BACKUP SET (≠00)?
NO
YES
PULL WHAT RTI WOULD PULL FROM STACK
PULL ONE (16 BIT) ADDRESS OFF STACK
IS ADDRESS = STRIP
NO
YES
PUT LAST ADDRESS BACK ON STACK
PRINT '*DEL*'
ENABLE IRQ
JUMP TO ADDRESS IN BACKUP

FIGURE 6

IRQ ROUTINE FLOWCHART

The non-maskable interrupt (NMI) is triggered by the falling edge of the one-shot that turns the power on. This occurs after the clock matches a time (typically the interrupt is triggered 8 ms after the processor has started the restart routine). The NMI usually indicates that warm up power to the sensors must be turned on or that a reading must be taken. The NMI routine (see figure 7) can be divided into three parts; preliminary initialization, determining the reason for the NMI and performing that function, and lastly, properly returning to the interrupted section of the restart routine.

The preliminary initialization finishes the initialization of clock b (the clock PIA, port B), and disables the clock board from further (or multiple) interrupts. A delay is performed (approximately 0.5 seconds) to allow sensors without warm up power to stabilize. Variables used in sequential readings are saved, and all variables are restored to their correct values. The present time (stored in memory) is then updated.

The NMI routine decides what should be done from the value of the status word (see figure 7). Before the NMI routine is finished the status word is changed to reflect what functions need to be performed next time. If all measurements have been taken, 68 hours and 15 minutes is loaded into the clock and all power is turned off. If measurements have not started yet and if they start today, their starting time is loaded into the clock. Otherwise 24 hours is loaded into the clock. (To allow a warm up time, on the days before the readings start NMI occurs at a warm up time before midnight.) All power is then turned off.

If the readings have already started, either warm up power is turned on or the appropriate readings are taken. If readings are taken, eight are averaged together for each port. After the readings are taken a check is performed for stopping time. If the stopping time has not been reached, the frequency of measurements minus the warm up time is loaded into the clock. If it wasn't warm up time all power is turned off; otherwise warm up power is left on.

Every time power is turned off a software delay (approximately 0.5 seconds) is performed. If the processor power switch is on, the delay has no purpose. If the switch is off, the delay makes certain that the processor executes no other instructions while power is turning off.

The final section of the NMI routine checks if the user options current readings, sequential readings, or dump was interrupted. If none of the above options were interrupted, a RTI is executed (the memory is still disabled). If dump was interrupted, the memory is enabled before the RTI. If sequential readings were interrupted, the current line of output is restarted on the next line. If current readings were interrupted, the current line of output is terminated.

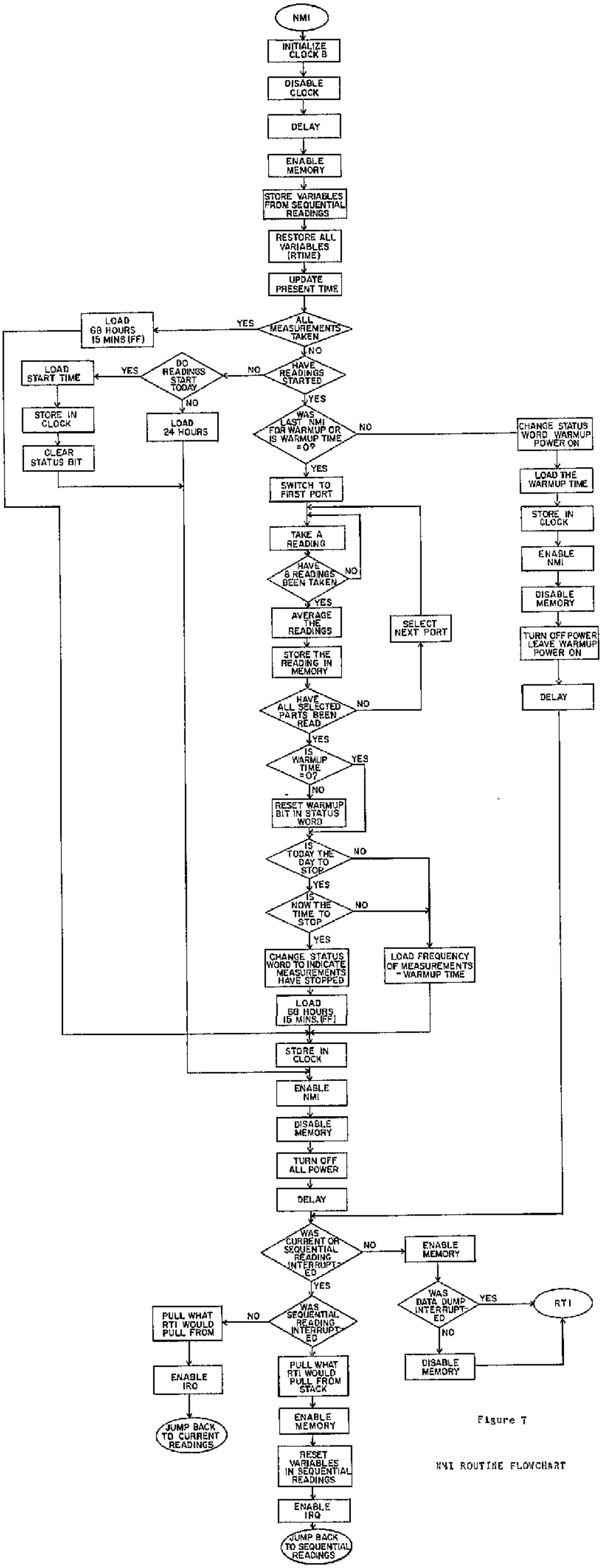

NMI
INITIALIZE CLOCK B
DISABLE CLOCK
DELAY
ENABLE MEMORY
STORE VARIABLES FROM SEQUENTIAL READINGS
RESTORE ALL VARIABLES (RTIME)
UPDATE PRESENT TIME
ALL MEASUREMENTS TAKEN
YES
LOAD 68 HOURS 15 MINS (FF)
NO
HAVE READINGS STARTED
NO
DO READINGS START TODAY
YES
LOAD START TIME
STORE IN CLOCK
CLEAR STATUS BIT
NO
LOAD 24 HOURS
YES
WAS LAST NMI FOR WARMUP OR IS WARMUP TIME =0?
NO
CHANGE STATUS WORD WARMUP POWER ON
LOAD THE WARMUP TIME
STORE IN CLOCK
ENABLE NMI
DISABLE MEMORY
TURN OFF POWER LEAVE WARMUP POWER ON
DELAY
YES
SWITCH TO FIRST PORT
TAKE A READING
HAVE 8 READINGS BEEN TAKEN
NO
YES
AVERAGE THE READINGS
STORE THE READING IN MEMORY
SELECT NEXT PORT
HAVE ALL SELECTED PARTS BEEN READ
NO
YES
IS WARMUP TIME =0?
YES
NO
RESET WARMUP BIT IN STATUS WORD
IS TODAY THE DAY TO STOP
NO
YES
IS NOW THE TIME TO STOP
NO
YES
CHANGE STATUS WORD TO INDICATE MEASUREMENTS HAVE STOPPED
LOAD FREQUENCY OF MEASUREMENTS - WARMUP TIME
LOAD 68 HOURS 15 MINS. (FF)
STORE IN CLOCK
ENABLE NMI
DISABLE MEMORY
TURN OFF ALL POWER
DELAY
WAS CURRENT OR SEQUENTIAL READING INTERRUPT-ED
NO
ENABLE MEMORY
WAS DATA DUMP INTERRUPT-ED
YES
RTI
NO
DISABLE MEMORY
YES
WAS SEQUENTIAL READING INTERRUPT-ED
NO
PULL WHAT RTI WOULD PULL FROM
ENABLE IRQ
JUMP BACK TO CURRENT READINGS
PULL WHAT RTI WOULD PULL FROM STACK
ENABLE MEMORY
RESET VARIABLES IN SEQUENTIAL READINGS
ENABLE IRQ
JUMP BACK TO SEQUENTIAL READINGS

Figure 7

NMI ROUTINE FLOWCHART

APPENDIX

CONNECTOR DESCRIPTIONS

A) TERMINAL AND PERIPHERAL CONNECTORS
AMPHENOL 25 PIN (FEMALE) D-SERIES BODY

PIN 2	TRANSMITTED DATA (even parity ASCII, RS232-C standard)
PIN 3	RECIEVED DATA (even parity ASCII, RS232-C standard)
PIN 4	REQUEST TO SEND
PIN 5	CLEAR TO SEND
PIN 6	DATA SET READY
PIN 7	GROUND

B) PORT (SENSOR) CONNECTORS
CANNON DEM5W1S WITH CENTER COAXIAL CONNECTION (RG/50)

PIN 1	-15 VOLT (WARM UP)
PIN 2	+15 VOLT (SWITCHED)
PIN 3	+15 VOLT (WARM UP)
PIN 4	-15 VOLT (SWITCHED)
COAXIAL	
CENTER	ANALOG INPUT (0-5 VOLTS)
SHEILD	GROUND

POWER SWITCHING BOARD
EDGE CONNECTOR SIGNALS

1	SWITCHED +12V FROM FRONT PANEL
2	SWITCHED +12V FROM FRONT PANEL
3	
4	POWER ON (1)
5	POWER OFF (2)
6	POWER OFF (3)
7	
8	
9	
10	GND
11	+12VDC
12	
13	
14	
15	GND
16	+5V
17	
18	+5V (SWITCHED)
19	+15V (SWITCHED)
20	-15V (SWITCHED)
21	-15V
22	+15V

MICROPROCESSOR BOARD
EDGE CONNECTOR SIGNALS

1	A0
A	A1
2	A2
B	A3
3	A4
C	A5
4	A6
D	A7
5	A8
E	A9
6	A10
F	A11
7	A12
H	A13
8	A14
J	$\overline{\text{A15}}$
9	D0
K	D1
10	D2
L	D3
11	D4
M	D5
12	D6
N	D7
13	$\text{R}/\overline{\text{W}}$
P	E
14	$\overline{\text{RESET}}$
R	$\overline{\text{NMI}}$
15	RESET (MANUAL)
S	RESET (MANUAL)
16	PIA SELECT (CLOCK)
T	PIA SELECT (A/D)
17	GND
U	R/W + MR
18	GND
V	
19	
W	+5V (SWITCHED)
20	E • VMA
Y	
21	+15V (TRANSMIT POWER)
Y	TRANSMIT SERIAL DATA
22	-15V (TRANSMIT POWER)
Z	RECEIVE SERIAL DATA FROM TERMINAL

CLOCK BOARD
EDGE CONNECTOR SIGNALS

1	A0
A	A1
2	MEMORY ENABLE
B	$\overline{\text{MEMORY ENABLE}}$
3	
C	
4	DATA DUMP
D	POWER ON (1)
5	
E	POWER OFF (2)
6	
F	POWER OFF (3)
7	
H	INITIATE CONVERSION (TO A/D BOARD)
8	
J	
9	D0
K	D1
10	D2
L	D3
11	D4
M	D5
12	D6
N	D7
13	R/$\overline{\text{W}}$
P	E
14	$\overline{\text{RESET}}$
R	$\overline{\text{NMI}}$
15	
S	
16	PIA SELECT (CLOCK)
T	
17	GND
U	
18	
V	+5V
19	
W	+5V (SWITCHED)
20	
X	
21	
Y	
22	
Z	

A/D BOARD
EDGE CONNECTOR SIGNALS

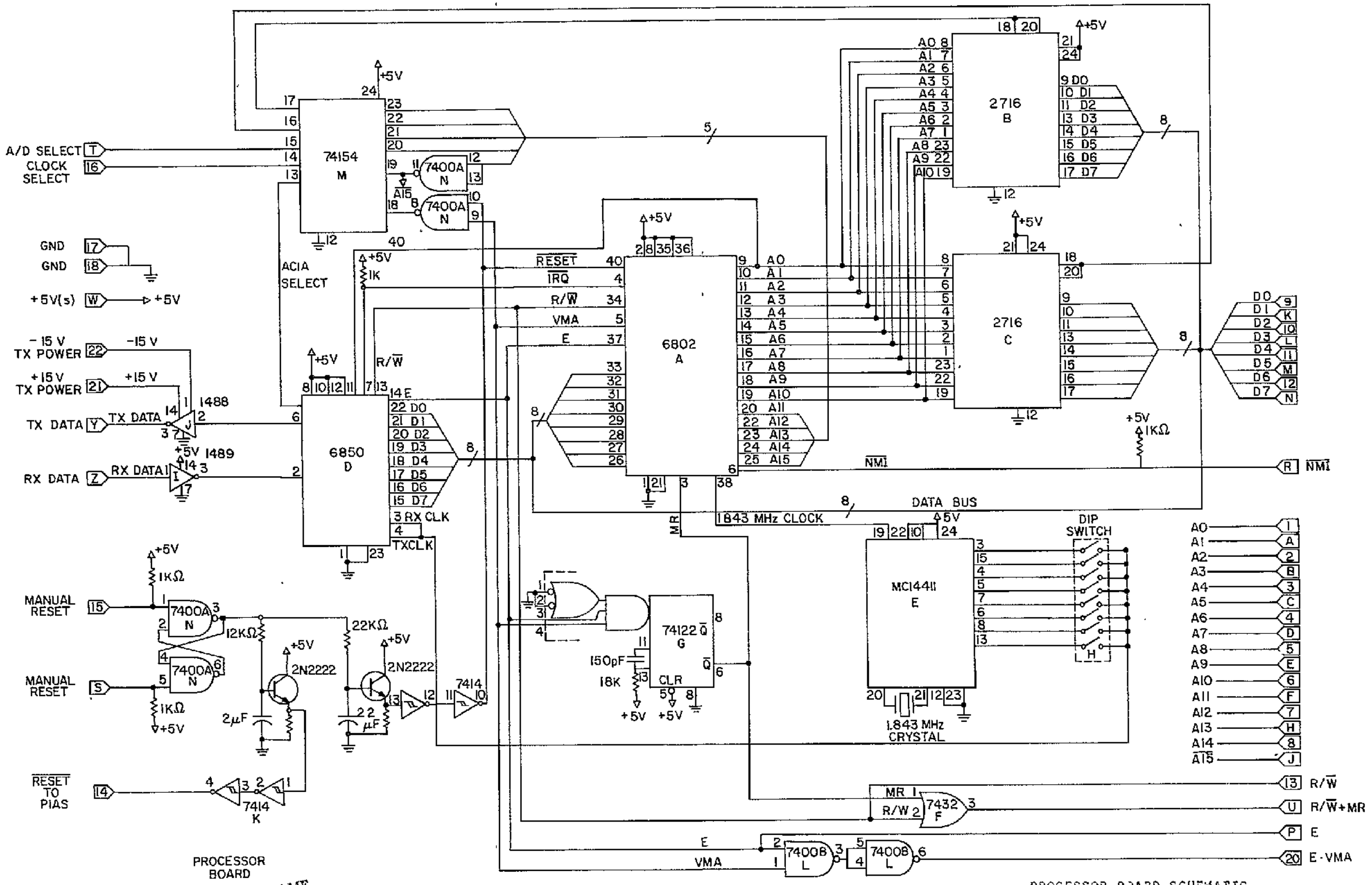
1	A0
A	A1
2	
B	
3	
C	
4	
D	
5	
E	
6	
F	
7	INITIATE CONVERSION
H	
8	
J	
9	D0
K	D1
10	D2
L	D3
11	D4
M	D5
12	D6
N	D7
13	R/$\overline{W}$
P	E
14	$\overline{RESET}$
R	
15	
S	
16	
T	PIA SELECT (A/D)
17	GND
U	
18	GND
V	
19	
W	+5V (SWITCHED)
20	
X	+15V (SWITCHED)
21	
Y	-15V (SWITCHED)
22	
Z	

MEMORY BOARDS
EDGE CONNECTOR SIGNALS

1	A0
A	A1
2	A2
B	A3
3	A4
C	A5
4	A6
D	A7
5	A8
E	A9
6	A10
F	A11
7	A12
H	A13
8	A14
J	$\overline{\text{A15}}$
9	D0
K	D1
10	D2
L	D3
11	D4
M	D5
12	D6
N	D7
13	R/$\overline{\text{W}}$ + MR
P	E • VMA
14	
R	
15	
S	
16	
T	MEMORY ENABLE
17	$\overline{\text{MEMORY ENABLE}}$
U	
18	GND
V	+5V
19	
W	
20	
X	
21	
Y	
22	
Z	

FRONT PANEL CONNECTOR

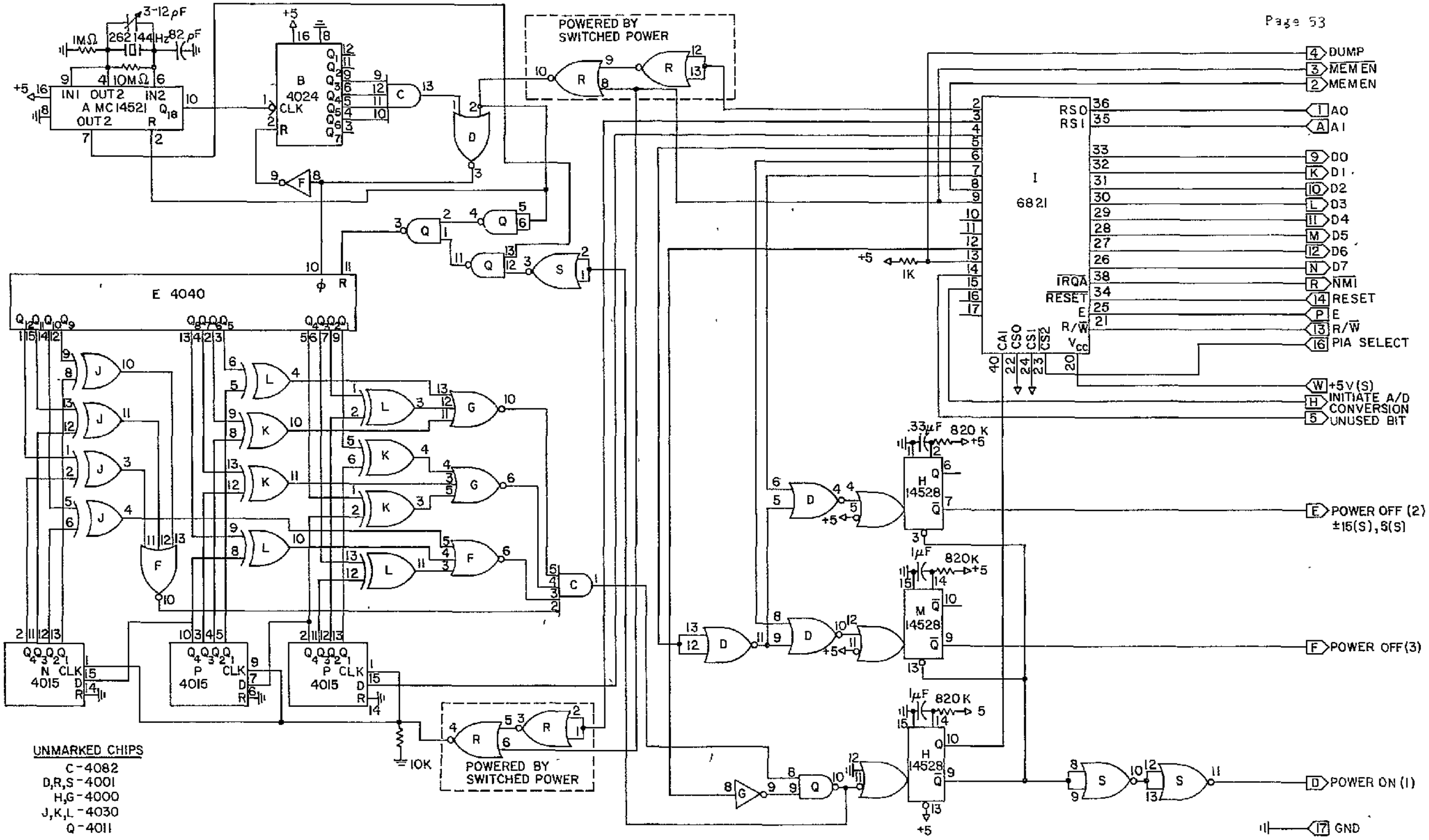
1	+12V (IN)
2	+12V (OUT)
3	+12V (IN)
4	+12V (OUT)
5	+15V (IN)
6	+15V (OUT)
7	-15V (IN)
8	-15V (OUT)
9	+15V (TRANSMIT POWER)
10	-15V (TRANSMIT POWER)
11	GND
12	RESET (MANUAL)
13	RESET (MANUAL)
14	DATA DUMP
15	REQUEST TO SEND (IN)
16	REQUEST TO SEND (OUT)
17	CLEAR TO SEND (IN)
18	CLEAR TO SEND (OUT)
19	DATA SET READY (IN)
20	DATA SET READY (OUT)
21	SERIAL DATA (FROM PROCESSOR)
22	SERIAL DATA (FROM PERIPHERAL)
23	SERIAL DATA (TO PERIPHERAL)
24	SERIAL DATA (TO TERMINAL)
25	GND



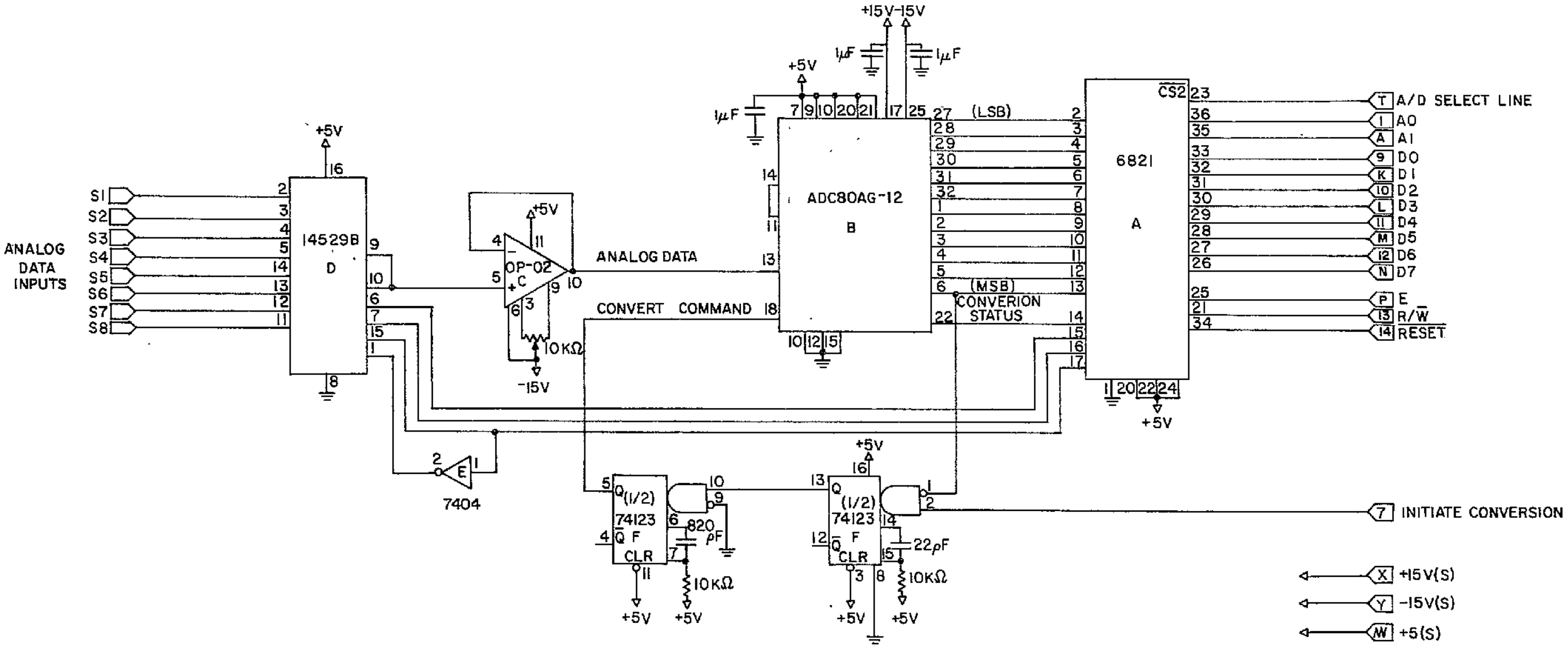
PROCESSOR BOARD SCHEMATIC

1 FOLDOUT FRAME

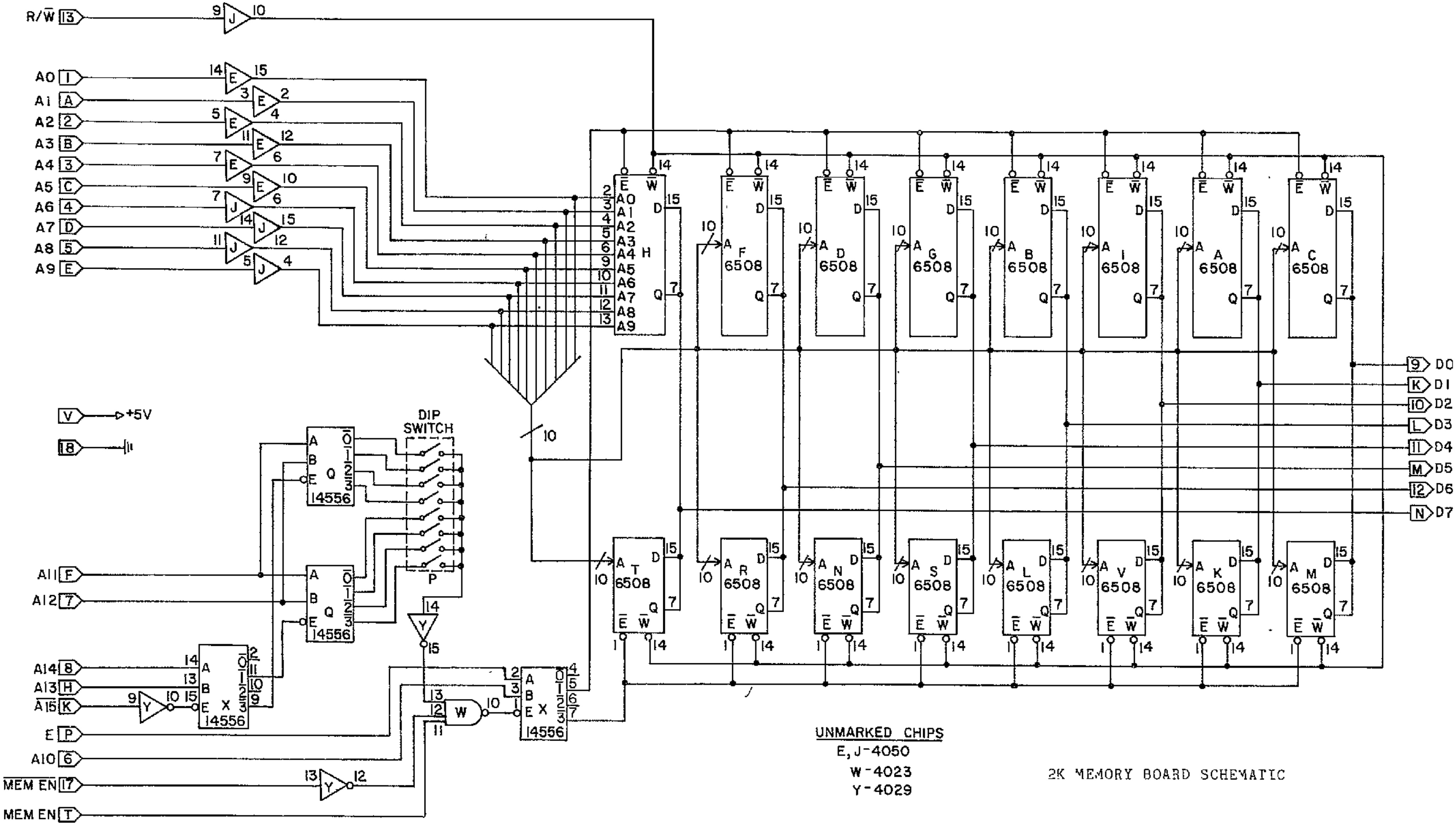
2 FOLDOUT FRAME



CLOCK BOARD SCHEMATIC



A/D CONVERSION BOARD SCHEMATIC

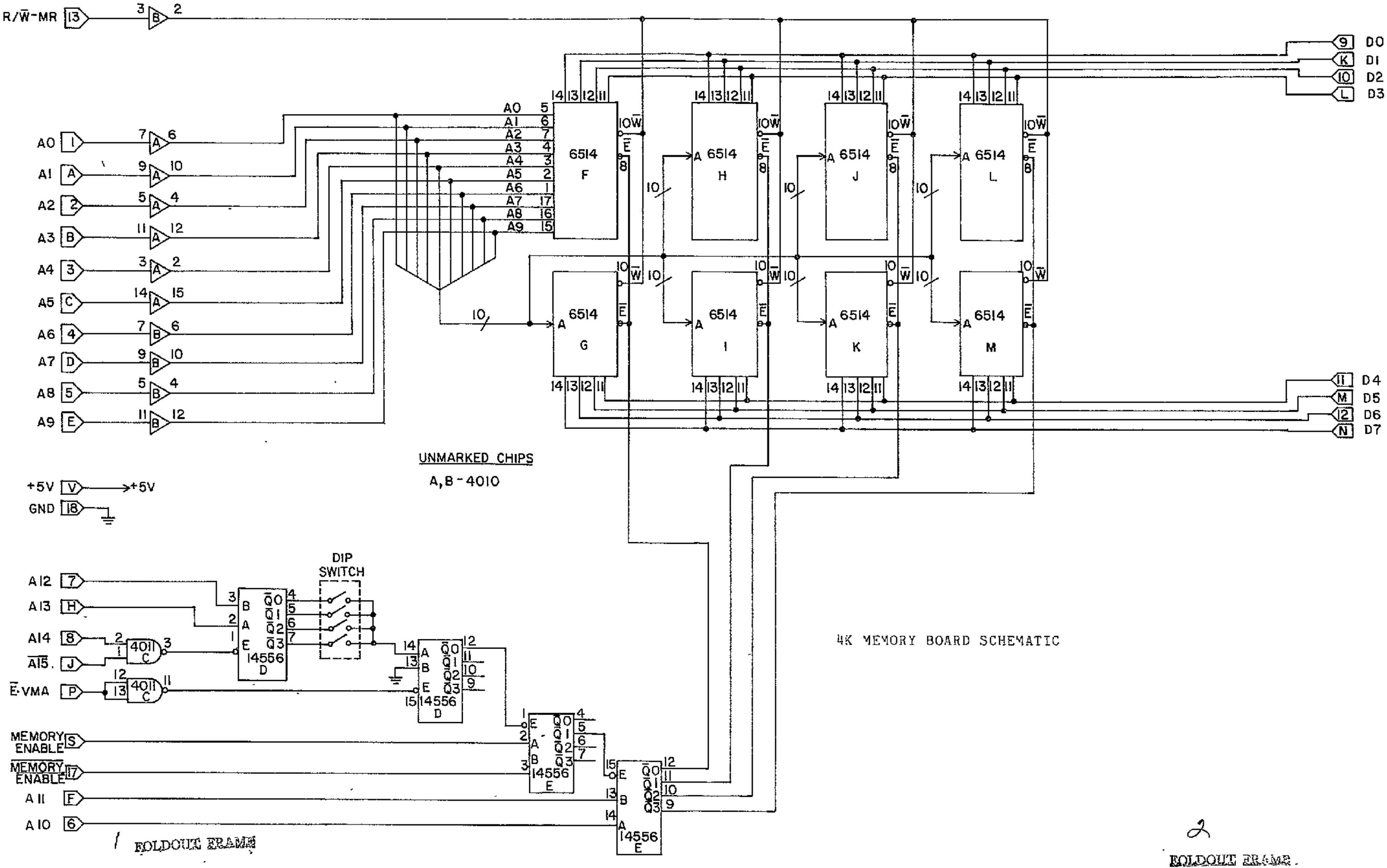


UNMARKED CHIPS
E, J-4050
W-4023
Y-4029

2K MEMORY BOARD SCHEMATIC

1 FOLDOUT FRAME

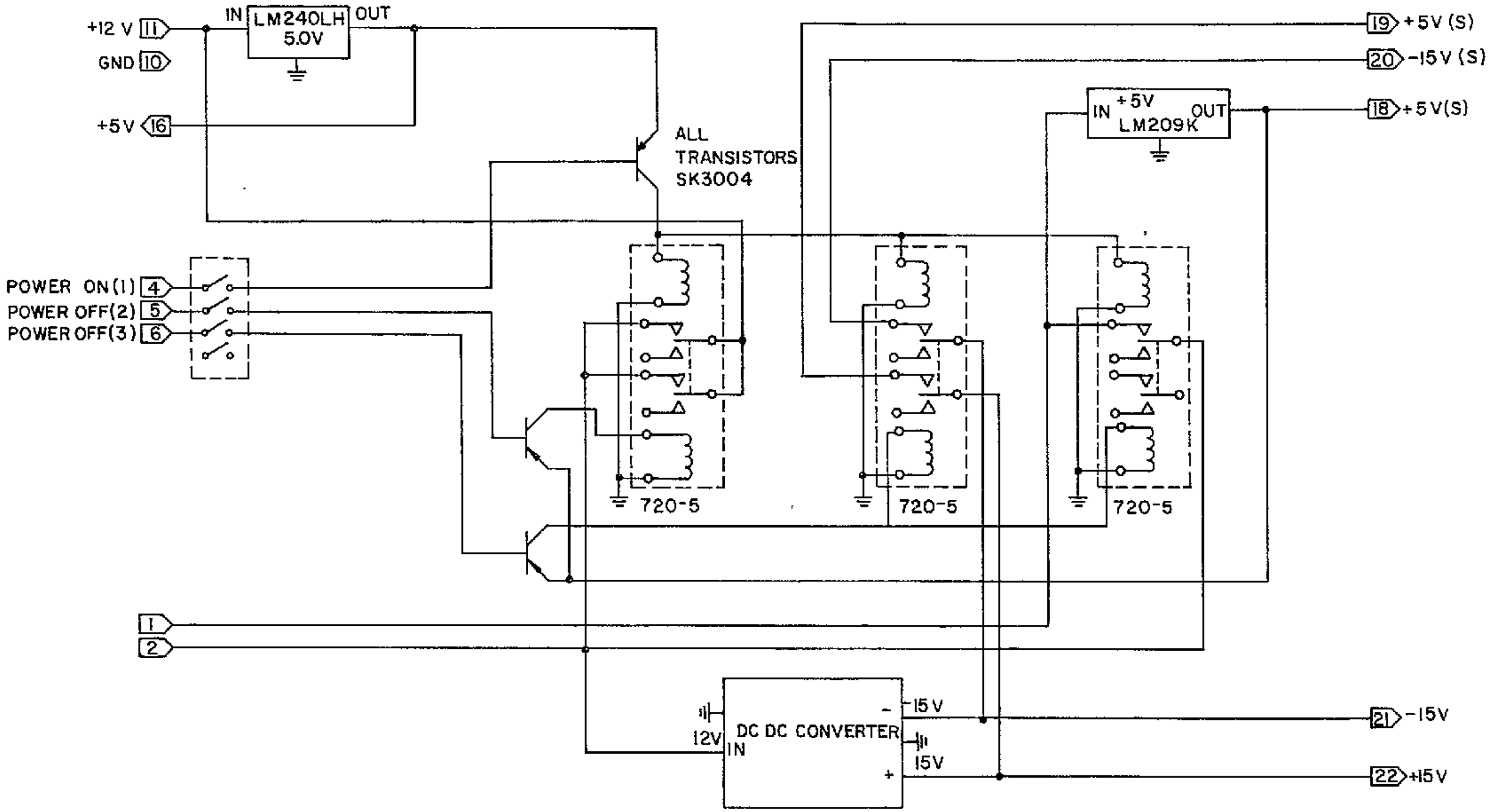
2 FOLDOUT FRAME



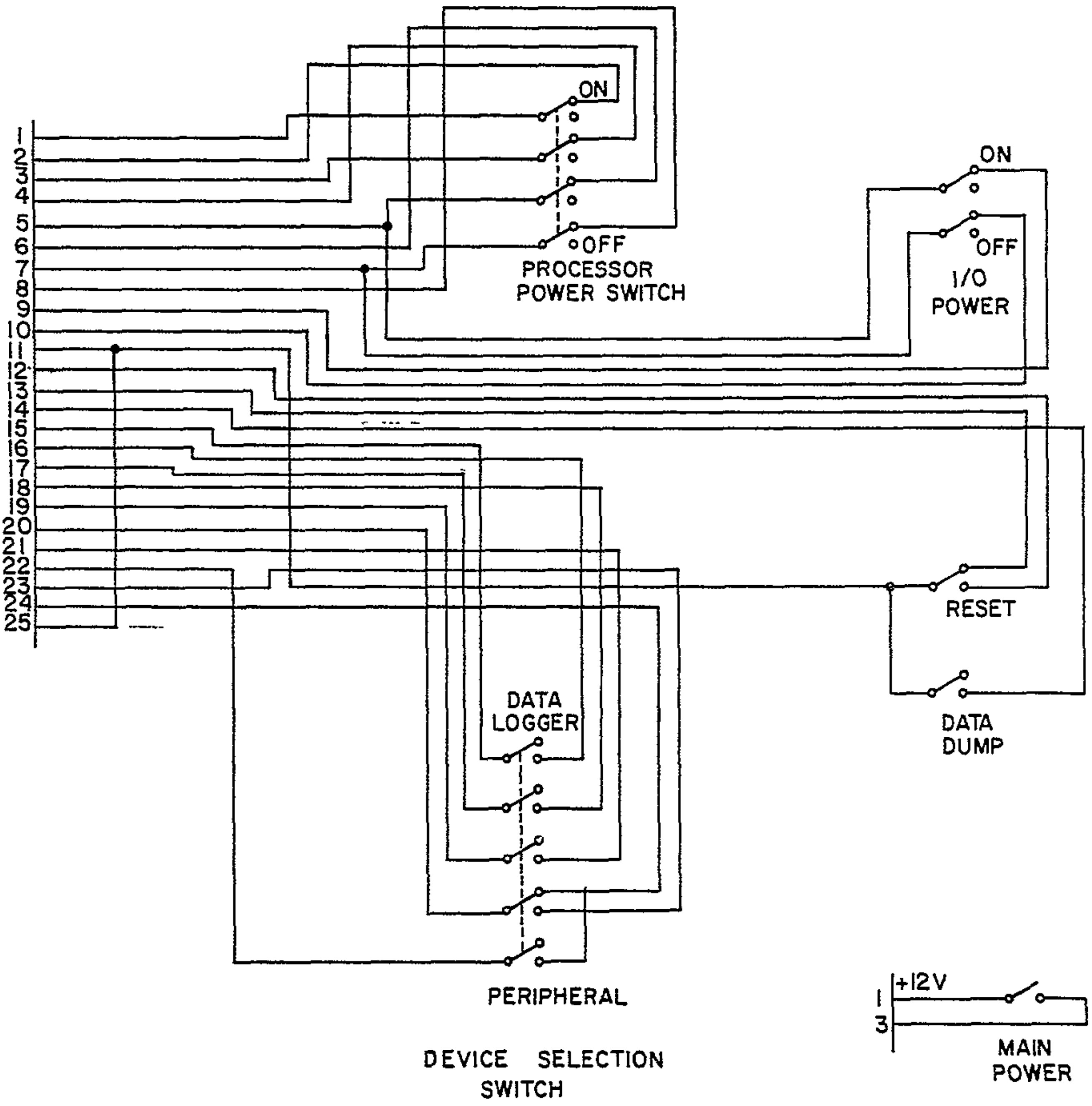
4K MEMORY BOARD SCHEMATIC

1 FOLDOUT FRAME

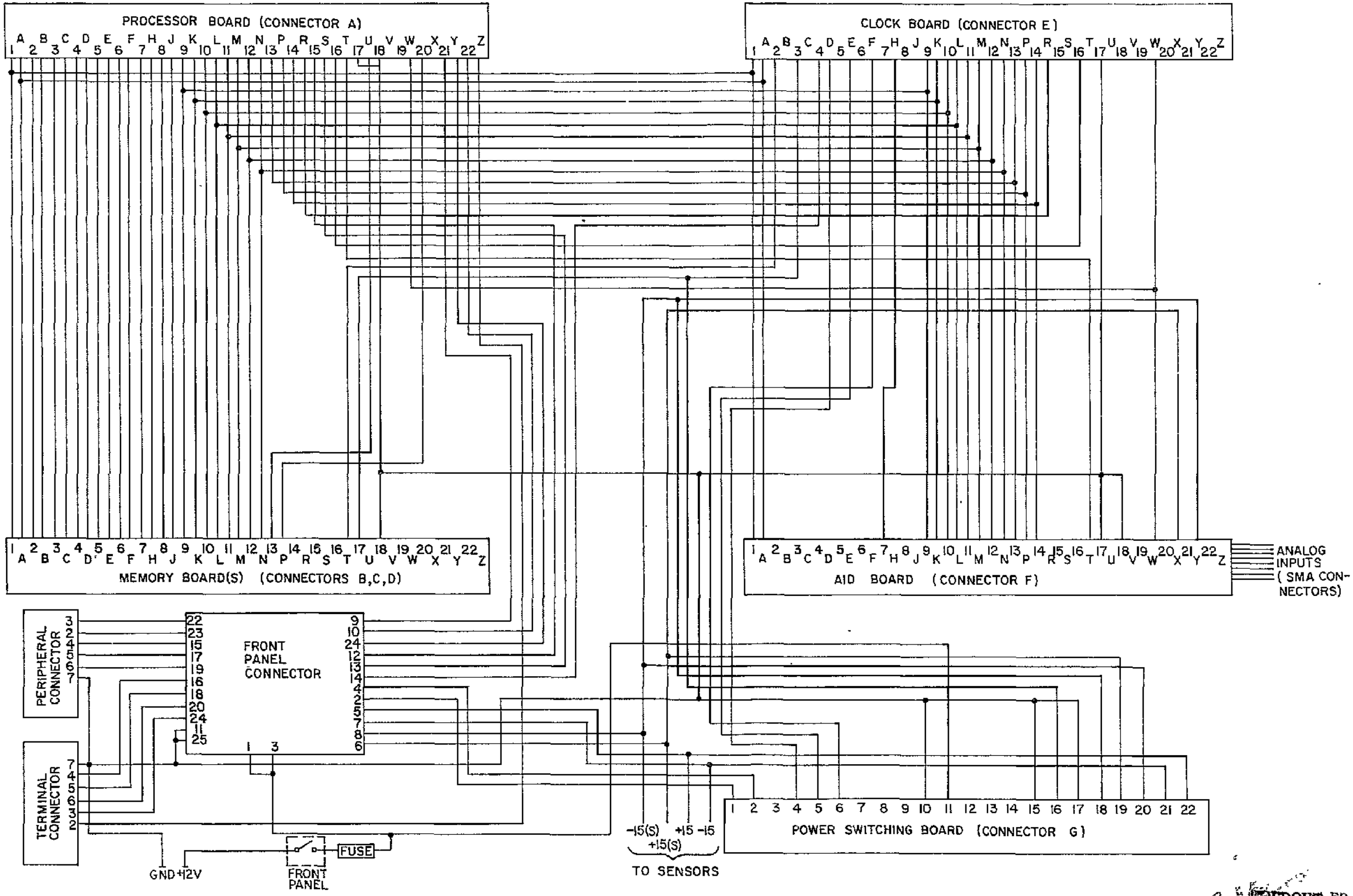
2 FOLDOUT FRAME.

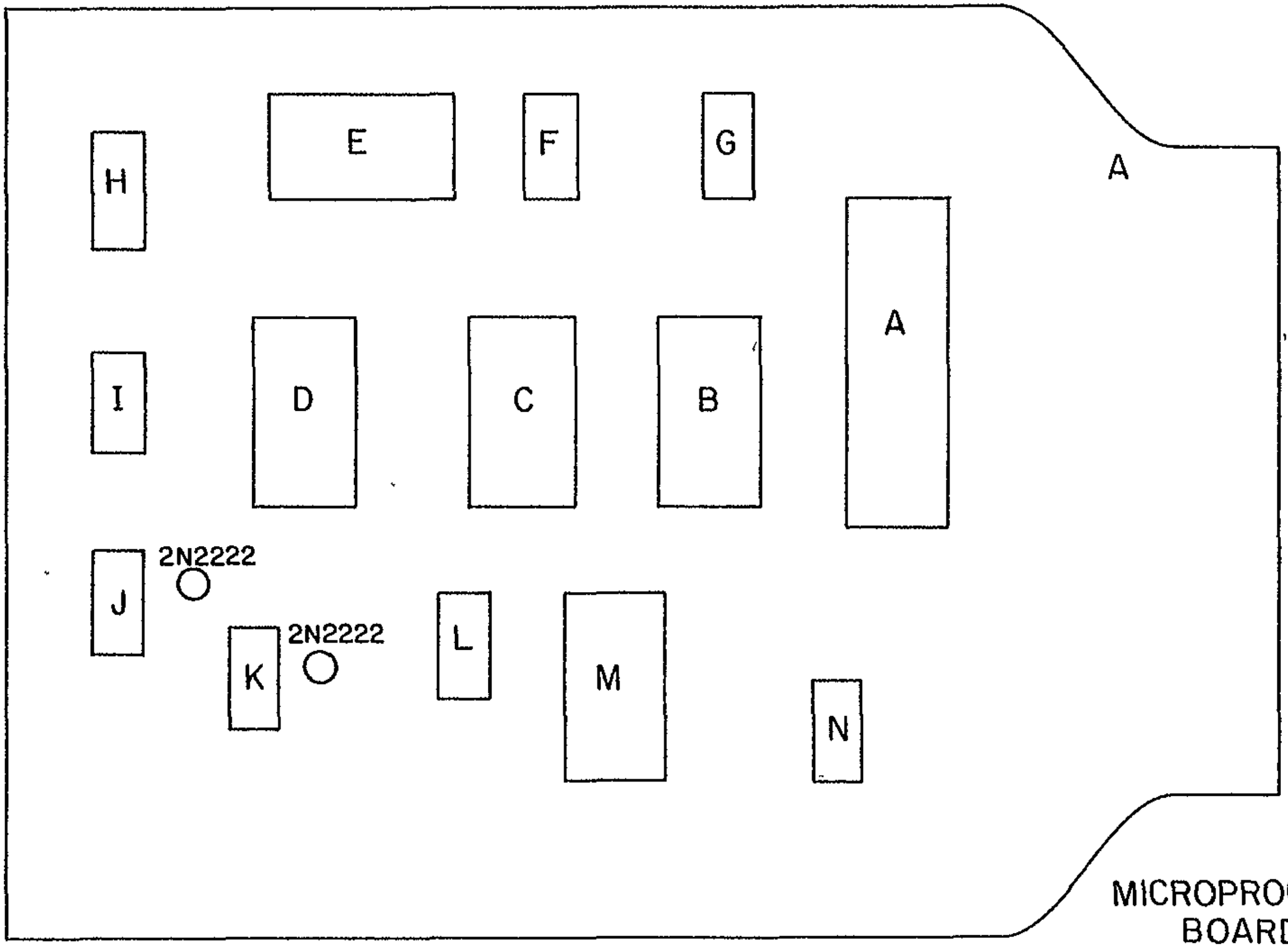


POWER SWITCHING BOARD SCHEMATIC

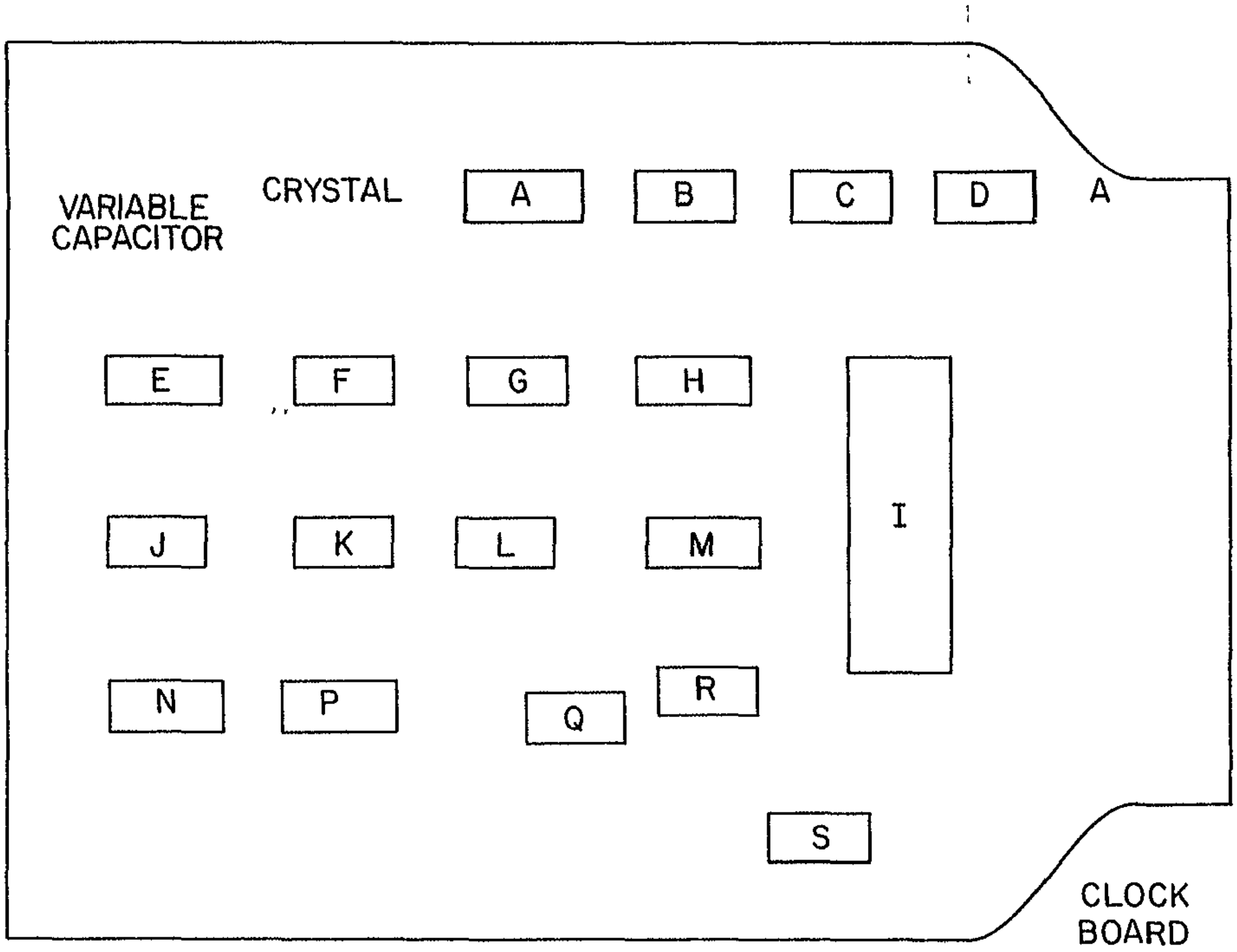


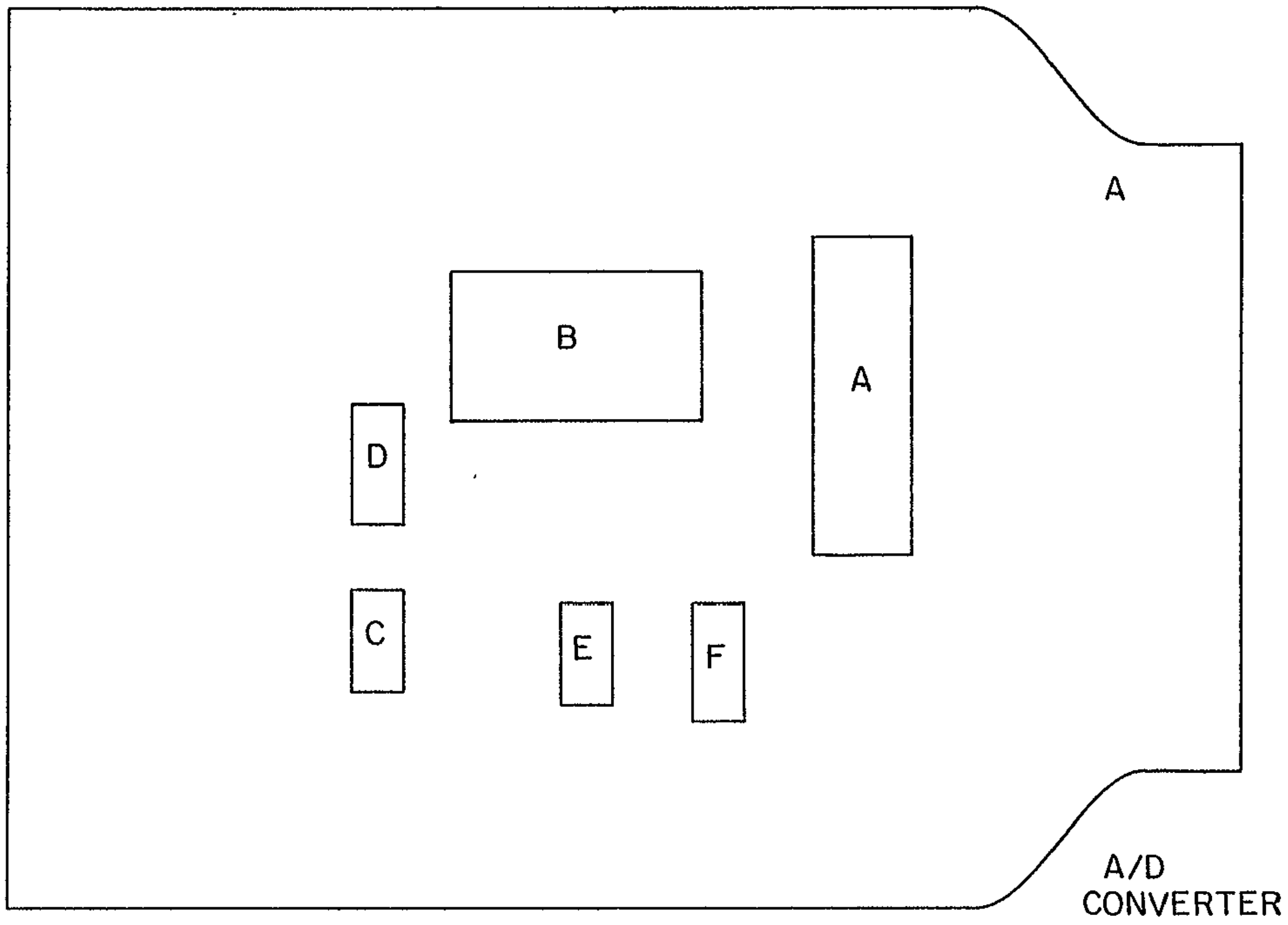
FRONT PANEL WIRING DIAGRAM

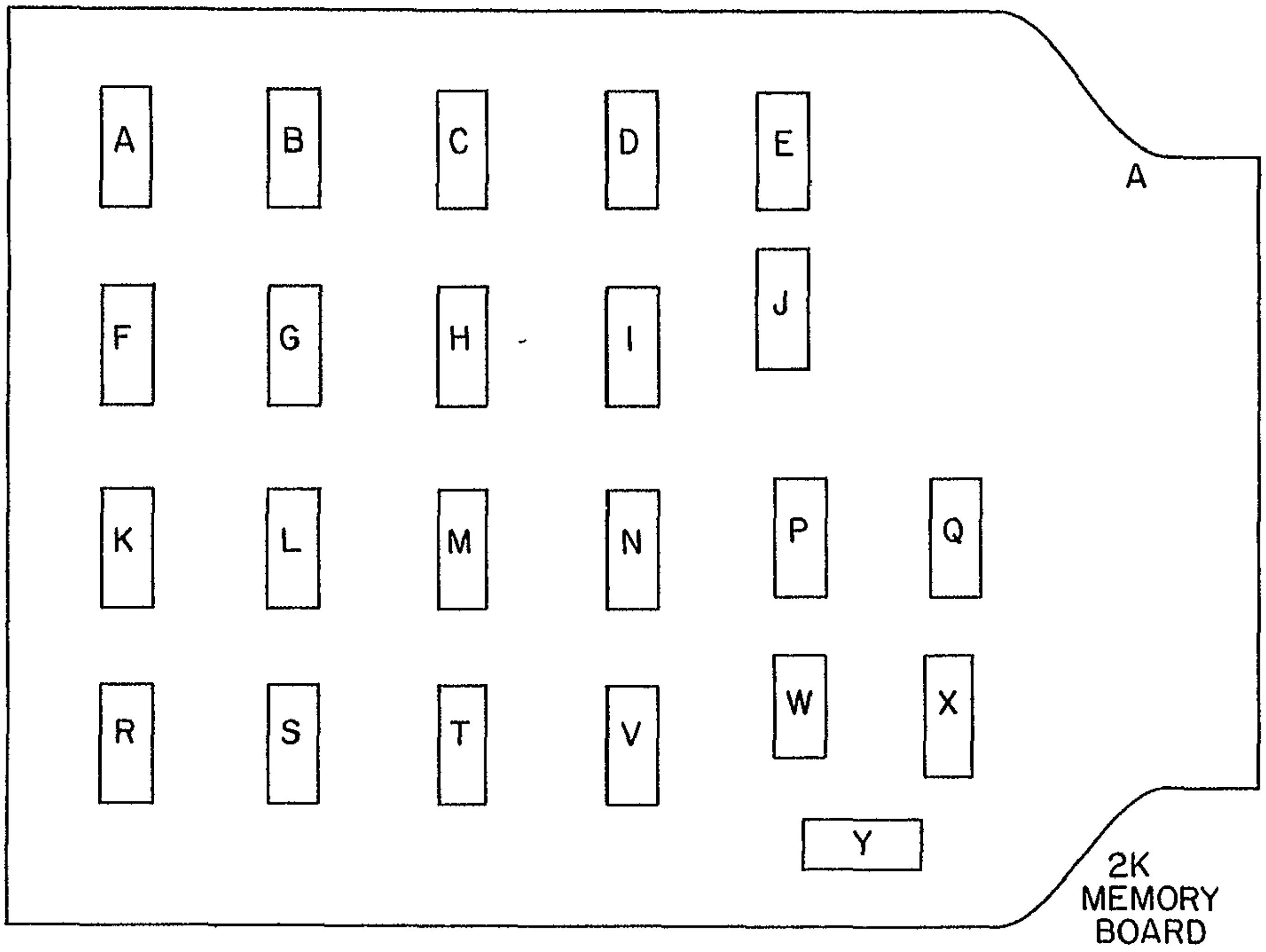

PROCESSOR BOARD (CONNECTOR A)
1 A 2 B 3 C 4 D 5 E 6 F 7 H 8 J 9 K 10 L 11 M 12 N 13 P 14 R 15 S 16 T 17 U 18 V 19 W 20 X 21 Y 22 Z
CLOCK BOARD (CONNECTOR E)
1 A 2 B 3 C 4 D 5 E 6 F 7 H 8 J 9 K 10 L 11 M 12 N 13 P 14 R 15 S 16 T 17 U 18 V 19 W 20 X 21 Y 22 Z
MEMORY BOARD(S) (CONNECTORS B,C,D)
A/D BOARD (CONNECTOR F)
ANALOG INPUTS (SMA CONNECTORS)
PERIPHERAL CONNECTOR
3 2 4 5 6 7
FRONT PANEL CONNECTOR
22 23 15 17 19 16 18 20 24 11 25
9 10 24 12 13 14 4 2 5 7 8 6
1 3
TERMINAL CONNECTOR
7 4 5 6 3 2
GND +12V
FRONT PANEL
FUSE
-15(S) +15 -15
+15(S)
TO SENSORS
POWER SWITCHING BOARD (CONNECTOR G)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22



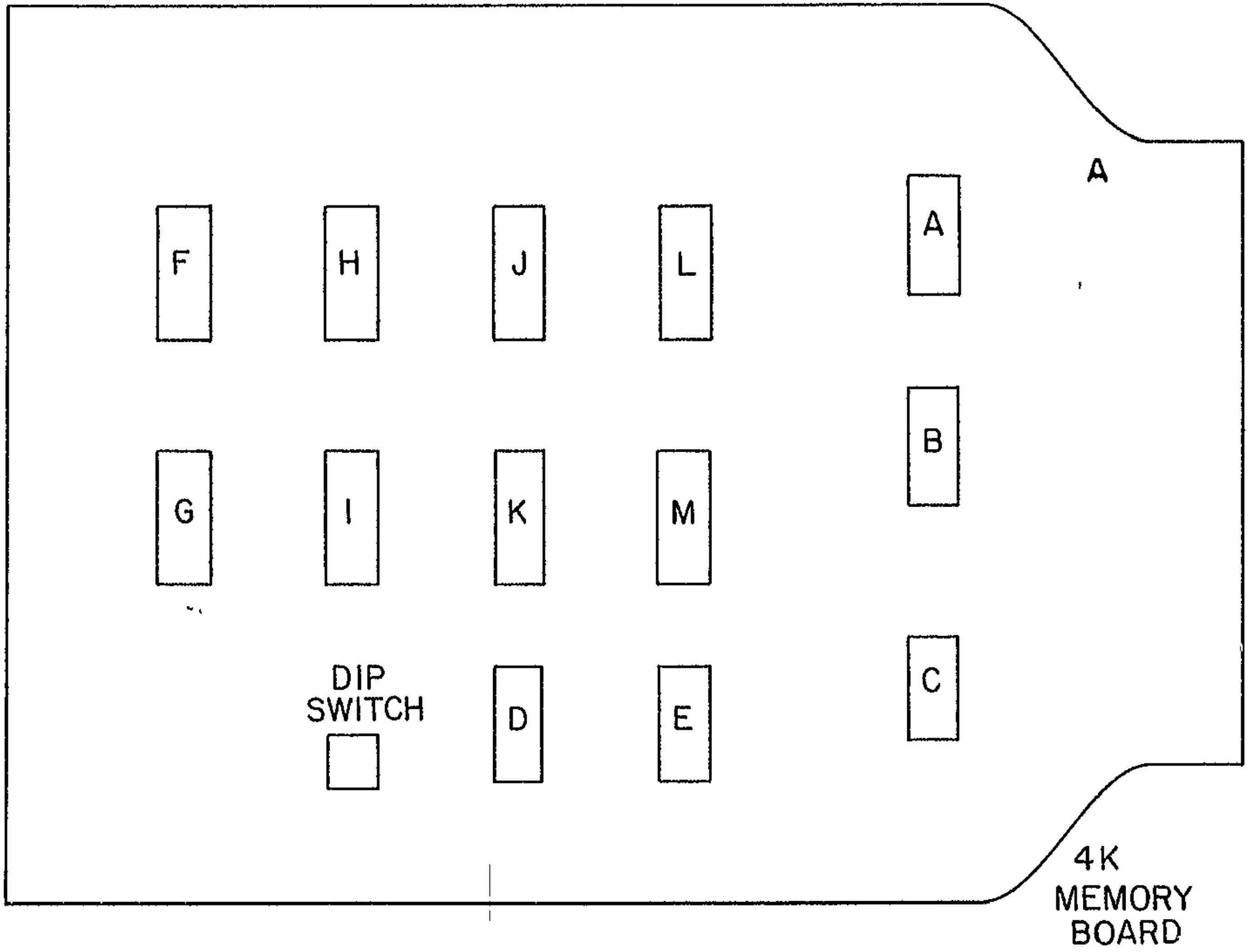
MICROPROCESSOR
BOARD


VARIABLE CAPACITOR
CRYSTAL
A
B
C
D
A
E
F
G
H
I
J
K
L
M
N
P
Q
R
S
CLOCK BOARD


A
B
A
D
C
E
F
A/D
CONVERTER



A
B
C
D
E
A
F
G
H
I
J
K
L
M
N
P
Q
R
S
T
V
W
X
Y
2K MEMORY BOARD


A
F
H
J
L
A
B
G
I
K
M
DIP
SWITCH
D
E
C
4K
MEMORY
BOARD

```
************************************
*                                  *
*                                  *
*                                  *
*                                  *
*           ↑THE ↑SOFTWARE         *
*                                  *
*                                  *
*                                  *
*                                  *
************************************
*
*
*
*
*↑THE FOLLOWING ARE ABREVIATIONS USED
*FREQUENTLY IN THE LISTING↑;
*
*       ↑A        ↑REGISTER ↑A
*
*       ↑B        ↑REGISTER ↑B
*
*       ↑I↑R       ↑THE ↑INDEX ↑REGISTER
*
*       ↑P↑I↑A      ↑PERIPHERAL ↑INTERFACE ↑ADAPTER
*               ↑THE ↑P↑I↑A IS A PARALLEL 16 BIT
*               ↑I/↑O PORT THAT ALSO HAS THE
*               CAPABILITIES TO DETECT INTERRUPTS.
*
*       ↑A↑C↑I↑A     ↑ASYNCHRONIS ↑COMMUNICATION ↑INTERFACE
*               ↑ADAPTER
*               ↑THE ↑A↑C↑I↑A HANDLES THE SERIAL TRANSMISSION
*               AND RECEIVING BETWEEN THE PROCESSOR AND
*               THE TERMINAL AND THE PERIPHERAL.
*
*       ↑N↑M↑I      ↑NON-↑MASKABLE ↑INTERRUPT
*               ↑THE ↑N↑M↑I IS CONNECTED TO THE CLOCK
*               BOARD SO THAT ANY TIME THE CLOCK TIMES
*               AN INTERVAL AN ↑N↑M↑I IS TRIGGERED WHEN
*               THE TIME INTERVAL IS REACHED.  ↑AN ↑I↑R↑Q
*               WILL BE LATCHED IF IT OCCURS DURING
*               AN ↑N↑M↑I, BUT IT WILL NOT BE EXECUTED
*               UNTIL THE ↑N↑M↑I  ROUTINE IS FINISHED.
*
*       ↑I↑R↑Q      ↑INTERRUPT ↑RE↑QUEST
*               ↑THE ↑I↑R↑Q IS CONNECTED TO THE ↑A↑C↑I↑A SO
*               THAT ANY TIME A CHARACTER IS RECEIVED
*               BY THE ↑A↑C↑I↑A AN ↑I↑R↑Q OCCURS.  ↑THE ↑I↑R↑Q
*               ROUTINE IS EXECUTED IMMEDIATELY UNLESS
*               A ↑N↑M↑I IS IN THE EXECUTION PROCESS.
*
*       [↑B↑A]     ↑BRACKETS INDICATE THAT THE PAIR OF
*               REGISTERS SHOULD BE CONSIDERED AS A
*               SINGLE 16 BIT NUMBER.  ↑THE FIRST
*               REGISTER IS THE HIGH ORDER 8 BITS AND
*               THE SECOND REGISTER FORMS THE LOW
*               ORDER 8 BITS.
```

```
********************************
*                              *
*                              *
*                              *
*          ↑VARIABLES          *
*             AND              *
*          ↑CONSTANTS          *
*                              *
*                              *
*                              *
********************************
*
*
*
*     RECEIV     ↑THE LAST WORD RECEIVED BY THE ↑A↑C↑I↑A IS
*                STORED HERE.
*
*     SIRG       ↑THE THIRD BIT OF THIS WORD IS SET WHEN-
*                EVER AN ↑I↑R↑Q OCCURS.
*
*     MONTH      ↑WHEN A MONTH IS ENTERED FROM THE TERMINAL
*                IT IS STORED HERE.
*
*     DAY        ↑WHEN A DAY IS ENTERED FROM THE TERMINAL
*                IT IS STORED HERE.
*
*     TIMEH      ↑WHEN A TIME IS ENTERED FROM THE TERMINAL
*                THE HIGH ORDER 8 BITS OF THE TIME IN
*                MINUTES IS STORED HERE.
*
*     TIMEL      ↑WHEN A TIME IS ENTERED FROM THE TERMINAL
*                THE LOW ORDER 8 BITS OF THE TIME IN
*                MINUTES IS STORED HERE.
*
*     PMONTH     ↑THE PRESENT MONTH.
*
*     PDAY       ↑THE PRESENT DAY.
*
*     PTIMEH     ↑THE 8 HIGH ORDER BITS OF THE PRESENT TIME.
*
*     PTIMEL     ↑THE 8 LOW ORDER BITS OF THE PRESENT TIME.
*
*     LMINH      ↑THE 8 HIGH ORDER BITS OF THE LAST TIME
*                STUFFED INTO THE CLOCK.
*
*     LMINL      ↑THE 8 LOW ORDER BITS OF THE LAST TIME
*                STUFFED INTO THE CLOCK.
*
*     TEMP1-13   ↑THIRTEEN TEMPORARY LOCATIONS.
*
*     SMONTH     ↑THE MONTH THE DATA LOGGER IS TO START
*                TAKING READINGS.
*
*     SDAY       ↑THE DAY THAT THE DATA LOGGER IS TO START
*                TAKING READINGS.
*
*     STIMEH     ↑THE 8 HIGH ORDER BITS OF THE TIME THAT
*               THE DATA LOGGER IS TO START TAKING READINGS.
*
*     STIMEL    ↑THE 8 LOW ORDER BITS OF THE TIME THAT
*               THE DATA LOGGER IS TO START TAKING READINGS.
*
*     HFREQ     ↑THE 8 HIGH ORDER BITS OF THE NUMBER OF
*               MINUTES BETWEEN READINGS.
*
*     LFREQ     ↑THE 8 LOW ORDER BITS OF THE NUMBER OF
*               MINUTES BETWEEN READINGS.
*
*     FMONTH    ↑THE MONTH THAT THE DATA LOGGER IS TO
*               STOP TAKING READINGS.
*
*     FDAY      ↑THE DAY THAT THE DATA LOGGER IS TO
*               STOP TAKING READINGS.
*
*     FTIMEH    ↑THE 8 HIGH ORDER BITS OF THE TIME THAT
*               THE DATA LOGGER IS TO STOP TAKING READINGS.
*
*     FTIMEL    ↑THE 8 LOW ORDER BITS OF THE TIME THAT
*               THE DATA LOGGER IS TO STOP TAKING READINGS.
*
*     CLOCKA    ↑THE ADDRESS OF PORT ↑A OF THE ↑P↑I↑A ON
*               THE CLOCK BOARD.
*
*     CLOCKB    ↑THE ADDRESS OF PORT ↑B OF THE ↑P↑I↑A ON
*               THE CLOCK BOARD.
*
*     WTIME     ↑THE WARMUP TIME IN MINUTES.
*
*     CONE,     ↑A CODED VERSION OF HOW MANY
*     CTWO,     DAYS, OVER 28, THERE ARE IN
*     CTHREE    EACH MONTH.
*
*     NUMAVR    ↑A CONSTANT, EQUAL TO EIGHT, THAT
*               INDICATES HOW MANY MEASUREMENTS TO
*               AVERAGE TOGETHER FOR ONE READING.
*
*     CONVH     ↑AFTER A MEASUREMENT IS TAKEN THE HIGH
*               ORDER 8 BITS OF THE MEASUREMENT IS STORED
*               IN THIS LOCATION.  ↑THIS LOCATION IS ALSO
*               USED AS THE SOURCE FOR THE CONVERSIONS
*               TO THE PROPER UNITS.
*
*     CONVL     ↑AFTER A MESUREMENT IS TAKEN THE LOW ORDER
*               8 BITS OF THE MEASUREMENT ARE STORED IN
*               THIS LOCATION.  ↑THIS LOCATION IS ALSO USED
*               AS THE SOURCE FOR THE CONVERSIONS TO THE
*               PROPER UNITS.
*
*     MDATAH    ↑THE HIGH ORDER 8 BITS OF THE NEXT LOCATION
*               THAT A READING IS STORED IN IS KEPT HERE.
*
*     MDATAL    ↑THE LOW ORDER 8 BITS OF THE NEXT LOCATION
*               THAT A READING IS STORED IN IS KEPT HERE.
*
```

```
*       STARTM     ↑THIS CONSTANT IS THE ADDRESS WHERE
*                  MEASUREMENTS ARE FIRST STORED IN MEMORY.
*
*       CNTRL      ↑THIS LOCATION IS A READ ONLY REGISTER IN
*                  THE ↑A↑C↑I↑A THAT HOLDS STATUS INFORMATION
*                  ABOUT THE LAST RECEIVED WORD.
*
*       DATA       ↑THIS LOCATION IS THE REGISTER IN THE
*                  ↑A↑C↑I↑A  THAT RECEIVES A WORD OR TRANSMITS
*                  A WORD.
*
*       EMASK      ↑THIS CONSTANT IS ↑A↑N↑DED WITH THE CNTRL WORD
*                  TO CHECK FOR A PARITY OR FRAME ERROR.
*
*       TMASK      ↑THE CONSTANT IS ↑A↑N↑DED WITH THE CNTRL WORD
*                  TO TELL IF THE ↑A↑C↑I↑A IS FREE TO TRANSMIT
*                  A WORD.
*
*
*       ADPIAA     ↑THIS IS THE LOCATION OF PORT ↑A OF THE
*                  ↑P↑I↑A ON THE ↑A/↑D BOARD.
*
*       ADPIAB     ↑THIS IS THE LOCATION OF PORT ↑B OF THE
*                  ↑P↑I↑A ON THE ↑A/↑D BOARD.
*
*       STATUS     ↑THIS WORD KEEPS STATUS INFORMATION THAT
*                  TELLS THE ↑N↑M↑I ROUTINE WHAT FUNCTION TO
*                  PERFORM NEXT.
*
*       TEMFIR,    ↑THESE LOCATIONS ARE USED TO TEMPORARILY
*       SECOND     SAVE THE CONTENTS OF THE ↑I↑R.
*
*       PORTBT     ↑THIS WORD TELLS WHICH ADDITIONAL PORTS
*                  HAVE BEEN SPECIFIED IN THE INITIALIZATION
*                  PROCEDURE.  ↑A ONE IN A BIT 0-7 SPECIFIES
*                  A PORT 1-8 RESPECTIVELY.
*
*       NPORTS     ↑THIS LOCATION HOLDS THE TOTAL NUMBER OF
*                  MEASUREMENTS THAT ARE TO BE TAKEN DURING
*                  EACH SET OF READINGS.
*
*       SENNUM     ↑THIS LOCATION TELLS SUBROUTINE ADCVAL
*                  WHICH PORT TO READ.
*
*       FULLR      ↑THIS LOCAION TEMPORARILY HOLDS THE NUMBER
*                  READINGS THAT ARE TO BE TAKEN.
*
*       INCIR      ↑THIS LOCCAION INDICATES HOW MANY MEMORY
*                  LOCATIONS ARE NEEDED TO STORE EACH SET
*                  OF MEASUREMENTS.
*
*       WHICH1,    ↑THESE LOCATIONS ARE USED IN DETERMINING
*       WHICH2     WHICH PORT IS TO BE READ NEXT.
*
*       CONVH1,    ↑THESE LOCATIONS ACT AS TEMPORARY LOCATIONS
*       CONVL1     FOR THE VALUE OF CONVH,CONVL.
*
```

```
*       NUMLOW,     ↑THESE LOCATIONS INDICATE HOW MANY READINGS
*       NUMHI       THE USER WOULD LIKE DISPLAYED DURING
*                   SEQUENTIAL READINGS.
*
*       BACKUP,     ↑THESE LOCATIONS HOLD AN ADDRESS TO JUMP
*       BACKP       BACK TO IF THE USER PRESSES THE ↑E↑S↑C↑A↑P↑E
*                   KEY WHEN HE IS INITIALIZING THE DATA LOGGER.
*
*       SAVINGS,    ↑THESE LOCATIONS ARE USED TO TEMPORARILY
*       SAVI        STORE THE CONTENTS OF THE ↑I↑R.
*
*
*
*
*
RECEIV  EGU     $12
SIRQ    EGU     $13
MONTH   EGU     $4001
DAY     EGU     $4002
TIMEH   EGU     $4003
TIMEL   EGU     $4004
PMONTH  EGU     $4005
PDAY    EGU     $4006
PTIMEH  EGU     $4007
PTIMEL  EGU     $4008
LMINH   EGU     $4009
LMINL   EGU     $400A
TEMP6   EGU     $400B
TEMP7   EGU     $400C
TEMP8   EGU     $400D
TEMP9   EGU     $400E
TEMP10  EGU     $400F
TEMP11  EGU     $4010
SMONTH  EGU     $4011
SDAY    EGU     $4012
STIMEH  EGU     $4013
STIMEL  EGU     $4014
HFREQ   EGU     $4015
LFREQ   EGU     $4016
FMONTH  EGU     $4017
FDAY    EGU     $4018
FTIMEH  EGU     $4019
FTIMEL  EGU     $401A
CLOCKA  EGU     $E000
CLOCKB  EGU     $E002
WTIME   EGU     $401B
TEMP1   EGU     $401C
TEMP2   EGU     $401D
TEMP3   EGU     $401E
TEMP4   EGU     $401F
TEMP5   EGU     $4020
TEMP12  EGU     $4021
TEMP13  EGU     $4022
CONE    EGU     $CE
CTWO    EGU     $EF
CTHREE  EGU     $BB
STKPTR  EGU     $7F
```

```
                    NUMAVR   EGU      $08
                    CONVH    EGU      $4023
                    CONVL    EGU      $4024
                    MDATAH   EGU      $4025
                    MDATAL   EGU      $4026
                    STARTH   EGU      $4040
                    CNTRL    EGU      $D800
                    EMASK    EGU      $70
                    DATA     EGU      $D801
                    TMASK    EGU      $02
                    ADPIAA   EGU      $E800
                    ADPIAB   EGU      $E802
                    STATUS   EGU      $4000
                    SDATE    EGU      $4027
                    TEMPIR   EGU      $4028
                    SECOND   EGU      $4029
                    TEMPI3   EGU      $02
                    TEMPI4   EGU      $03
                    PORTBT   EGU      $402A
                    NPORTS   EGU      $402B
                    SENNUM   EGU      $402C
                    FULLR    EGU      $402D
                    INCIR    EGU      $402E
                    WHICH1   EGU      $402F
                    WHICH2   EGU      $4030
                    CONVH1   EGU      $4031
                    CONVL1   EGU      $4032
                    NUMLOW   EGU      $4033
                    NUMHI    EGU      $4034
                    BACKUP   EGU      $00
                    BACKP    EGU      $01
                    SAVINGS  EGU      $14
                    SAVI     EGU      $15
                    SENSCV   EGU      $F800
                    MDELAY   EGU      $F9E5
                    RSHIFT4  EGU      $F94A
                    DELAY    EGU      $F9DE
                    *************************
                             CRG      $F000
F000     202020     SPACES   FCC      ↑7      ↑7
F003     2020
F005     00                  FCB      00
F006     444154     DATEQ    FCC      ↑7DATE↑/(MM/DD) ↑7
F009     453F20
F00C     4D4D2F
F00F     444429
F012     20
F013     00                  FCB      00
F014     54494D     TIMEQ    FCC      ↑7TIME↑/(HH↑$MM) ↑7
F017     453F28
F01A     48483A
F01D     4D4D29
F020     20
F021     00                  FCB      00
F022     535441     START    FCC      ↑7START ↑7
F025     525420
F028     00                  FCB      00
```

```
F029    53544F   STOP     FCC      ↑7STOP ↑7
F02C    5020
F02E    00                FCB      00
F02F    465245   FREQ     FCC      ↑7FREQUENCY OF ↑7
F032    515545
F035    4E4359
F038    204F46
F03B    20
F03C    4D4541            FCC      ↑7MEASUREMENTS↑7
F03F    535552
F042    454D45
F045    4E5453
F048    00                FCB      00
F049    3F2858   QFREQ    FCC      ↑7↑/(XXX) ↑7
F04C    585829
F04F    20
F050    00                FCB      00
F051    574152   WARMUP   FCC      ↑7WARMUP ↑7
F054    4D5550
F057    20
F058    00                FCB      00
F059    444154   DA       FCC      ↑7DATE ↑7
F05C    4520
F05E    00                FCB      00
F05F    54494D   TI       FCC      ↑7TIME ↑7
F062    4520
F064    00                FCB      00
F065    204D49   MIN      FCC      ↑7 MINS ↑7
F068    4E5320
F06B    00                FCB      00
F06C    4E4558   NEXTR    FCC      ↑7NEXT READING AT ↑7
F06F    542052
F072    454144
F075    494E47
F078    204154
F07B    20
F07C    00                FCB      00
F07D    504153   PR       FCC      ↑7PAST READINGS ↑7
F080    542052
F083    454144
F086    494E47
F089    5320
F08B    00                FCB      00
F08C    54454D   TE       FCC      ↑7TEMP ↑7
F08F    5020
F091    00                FCB      00
F092    54494D   WTIMEQ   FCC      ↑7TIME↑/(MM) ↑7
F095    453F28
F098    4D4D29
F09B    20
F09C    00                FCB      00
F09D    414444   ADDSEN   FCC      ↑7ADDITIONAL SENSOR↑/↑7
F0A0    495449
F0A3    4F4E41
F0A6    4C2053
F0A9    454E53
F0AC    4F523F
```

```
F0AF    00               FCB     00
F0B0    504F52  PORTB    FCC     ↑7PORT NUMBER↑/(1-8)↑7
F0B3    54204E
F0B6    554D42
F0B9    45523F
F0BC    283120
F0BF    3829
F0C1    00               FCB     00
F0C2    494E49  PINIT    FCC     ↑7INITIALIZATION↑/↑7
F0C5    544941
F0C8    4C495A
F0CB    415449
F0CE    4F4E3F
F0D1    00               FCB     00
F0D2    285945  YESNO    FCC     ↑7(YES,NO) ↑7
F0D5    532C4E
F0D8    4F2920
F0DB    0D0A7F           FCB     $0D,$0A,$7F,$7F,00
F0DE    7F00
F0E0    574152  WCURRS   FCC     ↑7WARM-UP TIME =↑7
F0E3    4D2D55
F0E6    502054
F0E9    494D45
F0EC    203D
F0EE    00               FCB     00
F0EF    545950  RS       FCC     /TYPE ↑7R↑7 FOR A READING, CONTROL/
F0F2    452027
F0F5    522720
F0F8    464F52
F0FB    204120
F0FE    524541
F101    44494E
F104    472C20
F107    434F4E
F10A    54524F
F10D    4C
F10E    202753           FCC     / ↑7S↑7 WHEN DONE/
F111    272057
F114    48454E
F117    20444F
F11A    4E45
F11C    0D0A7F           FCB     $0D,$0A,$7F,$7F,00
F11F    7F00
F121    524541  DONEIN   FCC     ↑7REASSIGN PORTS↑/↑7
F124    535349
F127    474E20
F12A    504F52
F12D    54533F
F130    00               FCB     00
                **********************
                *↑THIS SUBROUTINE WAITS FOR THE
                *INPUT OF A YES OR A NO.  ↑IF A
                *↑7YES↑7 IS TYPED A 0 IS RETURNED
                *IN ↑A. ↑IF ANYTHING ELSE IS TYPED
                *A 1 IS RETURNED IN ↑A.
F131    3E      CHYES    WAI                      ;WAIT FOR AN ↑I↑R↑Q
F132    9612             LDAA    RECEIV           ;↑A=LAST CHAR. RECEIVED
```

```
F134   8159               CMPA    ¬$↑7Y↑7
F136   2611               BNE     NOTY             ;BRANCH IF NOT A ↑7Y↑7
F138   3E                 WAI                      ;WAIT FOR THE SECOND CHAR
F139   9612               LDAA    RECEIV           ;↑A=LAST CHAR. RECEIVED
F13B   8145               CMPA    ¬$↑7E↑7
F13D   2608               BNE     NOTYES           ;BRANCH TO NOTYES IF NOT A ↑7E↑7
F13F   3E                 WAI                      ;WAIT FOR THE ↑7S↑7
F140   9612               LDAA    RECEIV           ;↑A=LAST CHAR RECEIVED
F142   8153               CMPA    ¬$↑7S↑7
F144   2604               BNE     NOTYES           ;BRANCH IF NOT A ↑7S↑7
F146   4F                 CLRA                     ;CLEAR ↑A MUST BE A YES
F147   2003               BRA     YESTST           ;BRANCH TO YES TEST
F149   3E        NOTY     WAI                      ;WAIT FOR ANOTHER CHAR
F14A   8601      NOTYES   LDAA    ¬$$01             ;NOT A YES LOAD ↑A=1
F14C   BDF555    YESTST   JSR     LFCR             ;SKIP A LINE
F14F   7F0012             CLR     RECEIV           ;CLEAR RECEIV
F152   4D                 TSTA                     ;TEST ↑A
F153   39                 RTS                      ;RETURN TO CALLER
                 ********************
                 **MULTIPLIES WANTS IN REG ↑A BY 10,
                 *LEAVING THE RESULT IN ↑A.
F154   48        AXTEN    ASLA                     ;CONTAINS 2≠↑X
F155   16                 TAB                      ;CONTAINS 2≠↑X
F156   48                 ASLA                     ;CONTAINS 4≠↑X
F157   48                 ASLA                     ;CONTAINS 8≠↑X
F158   1B                 ABA                      ;ADDS 8≠↑X AND 2≠↑X
F159   39                 RTS                      ;↑A=10≠↑X
                 ***********************
                 **INPUTS TWO DIGITS AND LEAVES
                 *THEIR DECIMAL VALUE IN ↑A.
F15A   3E        TWODIG   WAI                      ;WAIT FOR THE CHAR
F15B   9612               LDAA    RECEIV           ;MOVE THE CHAR INTO ↑A
F15D   BDF683             JSR     ASCHEX           ;CONVERT TO HEX EQUIVALENT
F160   BDF154             JSR     AXTEN            ;MULT ↑A BY 10
F163   16                 TAB                      ;STORE FIRST DIGIT IN ↑B
F164   3E                 WAI                      ;WAIT FOR NEXT DIGIT
F165   9612               LDAA    RECEIV           ;MOVE THE CHAR INTO ↑A
F167   BDF683             JSR     ASCHEX           ;CONVERT TO HEX EQUIVALENT
F16A   1B                 ABA                      ;ADD BOTH DIGITS
F16B   39                 RTS                      ;VALUE OF INPUT IN ↑A
                 ************************
                 **INPUTS ↑S↑T↑A↑R↑T,↑F↑I↑N↑I↑S↑H,AND ↑P↑R↑E↑S↑E↑N↑T
                 *DATE AND TIME;WARM-UP TIME AND
                 *FREQ. OF MEASUREMENTS.
F16C   CEF16C    INDAT    LDX     ¬$INDAT
F16F   DF00               STX     BACKUP           ;SET BACKUP FOR *DEL*
F171   BDF600             JSR     PSP
F174   BDF5F5             JSR     PSPACE
F177   BDF248             JSR     GETDAT           ;GETS THE PRESENT DATE AND TIME
F17A   BDF28D             JSR     RCLK             ;RESET THE CLOCK TO 0 TIME
F17D   CE4005             LDX     ¬$PMONTH          ;TELL REMEMBER WHERE TO STORE
F180   BDF278             JSR     REMEM            ;STORES PRESENT DATE AND TIME
F183   BDF555             JSR     LFCR             ;PRINTS LF AND CR
F186   CEF16C    INSTAR   LDX     ¬$INDAT
F189   DF00               STX     BACKUP
F18B   CEF022             LDX     ¬$START           ;LOAD START
F18E   BDF717             JSR     PRINT            ;PRINT ↑7START↑7
```

```
F191    BDF248            JSR     GETDAT          ;GET STARTING DATE AND TIME
F194    CE4011            LDX     ¬$SMONTH         ;TELL REMEMBER THIS IS STARTING
                                                  ;INFORMATION
F197    BDF276            JSR     REMEM           ;STORE THE STARTING TIME AND DATE
F19A    BDF555            JSR     LFCR            ;PRINT CR AND LF
F19D    CEF186   INSTOP   LDX     ¬$INSTAR
F1A0    DF00              STX     BACKUP          ;SET BACKUP FOR *DEL*
F1A2    CEF029            LDX     ¬$STOP           ;LOAD STOP
F1A5    BDF717            JSR     PRINT           ;PRINT ↑7STOP↑7
F1A8    BDF600            JSR     PSP
F1AB    BDF248            JSR     GETDAT          ;GET STOPPING DATE AND TIME
F1AE    CE4017            LDX     ¬$FMONTH         ;TELL REMEMBER TO STORE THIS AS
                                                  ;THE STOPPING INFORMATION
F1B1    BDF276            JSR     REMEM           ;STORE STOPPING DATE AND TIME
F1B4    BDF555            JSR     LFCR            ;PRINT LF AND CR
F1B7    CEF19D   INWARM   LDX     ¬$INSTOP
F1BA    DF00              STX     BACKUP          ;SET BACKUP FOR *DEL*
F1BC    CEF051            LDX     ¬$WARMUP         ;LOAD WARMUP
F1BF    BDF717            JSR     PRINT           ;PRINT ↑7WARMUP↑7
F1C2    CEF092            LDX     ¬$WTIMEQ         ;LOAD TIME↑/
F1C5    BDF717            JSR     PRINT           ;PRINT ↑7TIME↑/↑7
F1C8    BDF15A            JSR     TWODIG          ;GET TWO DIGITS
F1CB    B74016            STAA    WTIME           ;STORE THE WARMUP TIME
F1CE    BDF555            JSR     LFCR            ;PRINT LF AND CR
F1D1    CEF1B7   INFREQ   LDX     ¬$INWARM
F1D4    DF00              STX     BACKUP          ;SET BACKUP FOR *DEL*
F1D6    CEF02F            LDX     ¬$FREQ           ;LOAD FREQUENCY OF...
F1D9    BDF717            JSR     PRINT           ;PRINT ↑7FREQ. OF ....↑7
F1DC    CEF049            LDX     ¬$QFREQ          ;PRINT ↑/(XXX)
F1DF    BDF717            JSR     PRINT           ;PRINT IT
F1E2    BDF2AD            JSR     THRDIG          ;GET THREE DIGITS
F1E5    B74016            STAA    LFREQ           ;STORE LOW HALF OF FREQ.
F1E8    F74015            STAB    HFREQ           ;STORE HIGH HALF OF FREQ.
F1EB    CEF1D1   INPORT   LDX     ¬$INFREQ
F1EE    DF00              STX     BACKUP          ;SET UP BACKUP FOR *DEL*
F1F0    8603              LDAA    ¬$$03
F1F2    B7402B            STAA    NPORTS          ;¬$ OF READINGS NEED TO TAKE
F1F5    7F402A            CLR     PORTBT          ;CLEARS WHICH ADDITIONAL PORTS
F1F8    BDF555   MOREP    JSR     LFCR
F1FB    CEF09D            LDX     ¬$ADDSEN
F1FE    BDF717            JSR     PRINT           ;PRINT ↑7ADDITIONAL SENSOR..↑7
F201    CEF0D2            LDX     ¬$YESNO
F204    BDF717            JSR     PRINT           ;PRINT ↑7(YES,NO)↑/↑7
F207    BDF131            JSR     CHYES           ;WAIT FOR RESPONSE
F20A    2622              BNE     INDONE          ;IF NOT ZERO THEN DONE
F20C    CEF1EB            LDX     ¬$INPORT
F20F    DF00              STX     BACKUP          ;SETUP BACKUP FOR *DEL*
F211    CEF0B0            LDX     ¬$PORTB
F214    BDF717            JSR     PRINT           ;PRINT ↑7PORT NUMBER....↑7
F217    3E                WAI                     ;WAIT FOR RESPONSE
F218    9612              LDAA    RECEIV
F21A    BDF663            JSR     ASCHEX          ;CHANGE TO HEX
F21D    5F                CLRB                    ;CLEAR ↑B
F21E    0D                SEC                     ;SET CARRY = 1
F21F    59       ROLL     ROLB                    ;ROTATE LEFT ↑B
F220    4A                DECA                    ;DECREMENT ↑A
F221    26FC              BNE     ROLL            ;LOOP UNTIL ↑A=0
```

```
F223   FA402A           ORAB    PORTBT          ;OR ↑B WITH PORTBT
F226   F7402A           STAB    PORTBT          ;STORE THE RESULT
F229   7C402B           INC     NPORTS          ;INCREMENT THE NUMBER OF READINGS
F22C   20CA             BRA     MOREP           ;BRANCH BACK FOR MORE
F22E   BDF555   INDONE  JSR     LFCR            ;SKIP A LINE
F231   CEF121           LDX     ¬$DONEIN
F234   BDF717           JSR     PRINT           ;PRINT ↑7REASSIGN PORTS....↑7
F237   CEF002           LDX     ¬$YESNO
F23A   BDF717           JSR     PRINT
F23D   BDF131           JSR     CHYES           ;CHECK RESPONSE
F240   27A9             BEQ     INPORT          ;IF YES BRANCH BACK TO INPORT
F242   CE0000           LDX     ¬$0000
F245   DF00             STX     BACKUP          ;CLEAR BACKUP
F247   39               RTS                     ;RETURN TO CALLER
                ************************
                *↑GETS THE DATE AND TIME.
F248   CEF006   GETDAT  LDX     ¬$DATEQ          ;LOAD DATE↑/
F24B   BDF717           JSR     PRINT           ;PRINT ↑7DATE↑/↑7
F24E   BDF38C           JSR     DATE            ;INPUTS THE DATE
F251   CEF000           LDX     ¬$SPACES         ;LOAD SPACES
F254   BDF717           JSR     PRINT           ;PRINT FIVE SPACES
F257   CEF014           LDX     ¬$TIMEQ          ;LOAD TIME↑/
F25A   BDF717           JSR     PRINT           ;PRINT ↑7TIME↑/↑7
F25D   BDF39E           JSR     TIME            ;INPUTS THE TIME
F260   39               RTS
                ************************
                *↑COMPARES START DATE TO PRESENT
                *DATE, RETURNS A 0 IN ↑A IF NOT
                *THE SAME, 1 IF SAME.
F261   B64011   PSCOMP  LDAA    SMONTH          ;LOAD ↑A WITH STARTING MONTH
F264   B14005           CMPA    PMONTH          ;COMPARE ↑A TO THE PRESENT MONTH
F267   2702             BEQ     MSAME           ;IF EQUAL BRANCH TO MSAME
F269   4F               CLRA                    ;IF NOT EQUAL CLEAR ↑A
F26A   39               RTS                     ;AND RETURN VALUE ↑A=0
F26B   B64012   MSAME   LDAA    SDAY            ;LOAD ↑A WITH STARTING DAY
F26E   B14006           CMPA    PDAY            ;COMPARE ↑A TO THE PRESENT DAY
F271   2702             BEQ     DSAME           ;BRANCH IF THE SAME TO DSAME
F273   4F               CLRA                    ;IF NOT THE SAME CLEAR ↑A
F274   39               RTS                     ;AND RETURN VALUE ↑A=0
F275   8601     DSAME   LDAA    ¬$$01            ;THE DATES ARE THE SAME
F277   39               RTS                     ;RETURN VALUE ↑A=1
                ******************
                *↑STORES THE NUMBERS IN MONTH, DAY,
                *TIMEH, AND TIMEL INTO 4 CONTIGUOUS
                *LOCATIONS.  ↑THE ADDRESS OF THE FIRST
                *LOCATION MUST BE IN THE INDEX REG.
                *BEFORE THE SUBROUTINE IS CALLED.
F278   B64001   REMEM   LDAA    MONTH           ;LOAD MONTH INTO ↑A
F27B   A700             STAA    00,X            ;STORE INTO FIRST LOCATION
F27D   B64002           LDAA    DAY             ;LOAD DAY INTO ↑A
F280   A701             STAA    01,X            ;STORE DAY INTO SECOND LOCATION
F282   B64003           LDAA    TIMEH           ;LOAD TIMEH INTO ↑A
F285   A702             STAA    02,X            ;STORE TIMEH INTO THIRD LOCATION
F287   B64004           LDAA    TIMEL           ;LOAD TIMEL INTO ↑A
F28A   A703             STAA    03,X            ;STORE TIMEL INTO FOURTH LOCATION
F28C   39               RTS                     ;RETURN
                *********************
                *↑THIS SUBROUTINE RESETS THE CLOCK
                *TO TIME 0.
F28D    36      RCLK    PSHA                    ;SAVE ↑A
F28E    37              PSHB                    ;SAVE ↑B
F28F    B6E000          LDAA    CLOCKA          ;LOAD CLOCKA INTO ↑A
F292    16              TAB                     ;COPY ↑A INTO ↑B
F293    8A01            ORAA    ¬$$01           ;↑O↑R ↑A WITH 01 (HEX)
F295    847F            ANDA    ¬$$7F           ;↑A↑N↑D ↑A WITH 7F (HEX)
F297    B7E000          STAA    CLOCKA  -       ;STORE ↑A IN CLOCKA
F29A    F7E000          STAB    CLOCKA          ;RETURN CLOCKA TO ORIGINAL VALUE
F29D    33              PULB                    ;RESTORE ↑B
F29E    32              PULA                    ;RESTORE ↑A
F29F    39              RTS
                **********************
                *↑O↑U↑T↑A TRANSMITS TO THE OUTPUT DEVICE
                *(TERMINAL OR TELETYPE) THE CONTENTS
                *OF REGISTER ↑A.
F2A0    37      OUTA    PSHB                    ;SAVE ↑B
F2A1    F6D800  BACKO   LDAB    CNTRL           ;LOAD ↑B WITH ↑A↑C↑I↑A CONTROL WORD
F2A4    C402            ANDB    ¬STMASK         ;CHECK BIT 1
F2A6    27F9            BEQ     BACKO           ;WAIT IF BIT IS 0
F2A8    B7D801  FORWORD STAA    DATA            ;WRITE OUT ↑A
F2AB    33              PULB                    ;RESTORE ↑B
F2AC    39              RTS
                **********************
                **********************
                *↑INPUTS THREE DIGITS INTO [↑A↑B].
F2AD    3E      THRDIG  WAI                     ;WAIT FOR FIRST DIGIT
F2AE    9612            LDAA    RECEIV          ;LOAD DIGIT INTO ↑A
F2B0    BDF683          JSR     ASCHEX          ;CONVERT TO HEX
F2B3    BDF154          JSR     AXTEN           ;MULTIPLY BY TEN
F2B6    B7401C          STAA    TEMP1           ;STORE IN TEMP1
F2B9    5F              CLRB                    ;CLEAR ↑B
F2BA    CE0009          LDX     ¬$9             ;PREPARE TO MULT. TEN TIMES
F2BD    BB401C  TENTIM  ADDA    TEMP1           ;ADD TEMP1 TO SELF NINE TIMES
F2C0    C900            ADCB    ¬$00            ;A 2 REGISTER ADD,ADD CARRY
F2C2    09              DEX                     ;DECREMENT LOOP COUNTER
F2C3    26F8            BNE     TENTIM          ;BRANCH UNTIL ADDED 9 TIMES
F2C5    B7401C          STAA    TEMP1           ;STORE VALUE OF FIRST DIGIT
F2C8    F7401D          STAB    TEMP2           ;STORE LOWER HALF OF DIGIT
F2CB    BDF15A          JSR     TWODIG          ;GET THE OTHER TWO DIGITS
F2CE    5F              CLRB                    ;CLEAR ↑B
F2CF    BB401C          ADDA    TEMP1           ;ADD LOWER HALF OF FIRST DIGIT
F2D2    F9401D          ADCB    TEMP2           ;ADD UPPER HALF OF FIRST DIGIT
F2D5    39              RTS                     ;DECIMAL VALUE IN [↑A↑B].
                ********************
                *↑MIDNIGHT MINUS PRESENT TIME PUT IN [↑A↑B].
F2D6    8605    MIDNIG  LDAA    ¬$$05           ;24^;00 = 05↑A0 IN HEX
F2D8    C6A0            LDAB    ¬$$A0           ;LOAD THIS INTO [↑A↑B]
F2DA    F04008          SUBB    PTIMEL          ;SUBTRACT LOWER HALF OF PRESENT TIME
F2DD    B24007          SBCA    PTIMEH          ;SUBTRACT UPPER HALF OF PRESENT TIME
F2E0    39              RTS
                ********************
                *↑UPDATES THE PRESENT TIME, DAY, AND MONTH BASED
                *ON THE LAST TIME IN THE CLOCK (LTIMEH,LTIMEL).
F2E1    B64008  UPTIME  LDAA    PTIMEL
F2E4    F64007          LDAB    PTIMEH
```

```
F2E7    BB400A             ADDA     LMINL           ;ADD LAST TIME IN CLOCK
F2EA    F94009             ADCB     LMINH           ;BOTH LOW AND HIGH ORDER
F2ED    B74008             STAA     PTIMEL
F2F0    F74007             STAB     PTIMEH          ;STORE BACK IN MEMORY
F2F3    4C                 INCA                     ;PROVIDES A CHECK FOR LOWCHK
F2F4    C105               CMPB     ¬$$05            ;IS IT EQUAL TO HIGH ORDER OF ONE DAY
F2F6    2703               BEQ      LOWCHK          ;IF YES GO CHECK LOW HALF
F2F8    2E06               BGT      FIXDAY          ;IF GREATER BRANCH TO FIX DAY
F2FA    39                 RTS                      ;RETURN TO CALLER
F2FB    81A0      LOWCHK   CMPA     ¬$$A0            ;IS LOW HALF = LOW HALF ON ONE DAY
F2FD    2201               BHI      FIXDAY          ;IF LARGER GO TO FIXDAY
F2FF    39                 RTS                      ;RETURN TO CALLER
F300    80A1      FIXDAY   SUBA     ¬$$A1
F302    C205               SBCB     ¬$$05            ;SUBTRACT 24 HOURS (IN MINS.)
F304    B74008             STAA     PTIMEL
F307    F74007             STAB     PTIMEH          ;STORE BACK INTO PTIMEH+L
F30A    B64006             LDAA     PDAY
F30D    4C                 INCA                     ;INCREMENT THE DAY
F30E    B74006             STAA     PDAY            ;STORE THE DAY
F311    811C               CMPA     ¬$$1C            ;COMPARE TO 28 DAYS
F313    2C01               BGE      MONCHK          ;IF GREATER THAN GOTO MONCHK
F315    39                 RTS                      ;RETURN TO CALLER
F316    C6CE      MONCHK   LDAB     ¬$CONE
F318    F7401D             STAB     TEMP2
F31B    C6EF               LDAB     ¬$CTWO
F31D    F7401E             STAB     TEMP3
F320    C6BB               LDAB     ¬$CTHREE
F322    F7401F             STAB     TEMP4
F325    F64005             LDAB     PMONTH          ;*B=MONTH
F328    5A        CSHIFT   DECB                     ;DECREMENT THE MONTH
F329    2D08               BLT      MONDAY          ;IF ZERO DONE SHIFTING
F32B    BDF34B             JSR      TEMPSH
F32E    BDF34B             JSR      TEMPSH          ;SHIFT CONE,CTWO,CTHREE TWICE
F331    20F5               BRA      CSHIFT          ;LOOP BACK
F333    F6401C    MONDAY   LDAB     TEMP1
F336    C403               ANDB     ¬$$03            ;GET OUT THE NUMBER OF DAYS
                                                    ;OVER 28 THAT ARE IN THIS MONTH
F338    CB1C               ADDB     ¬$28             ;ADD 28 TO GET DAYS IN MONTH
F33A    11                 CBA                      ;COMPARE DAY TO DAYS IN MONTH
F33B    2E01               BGT      MONFIX          ;IF GREATER THAN GOTO MONFIX
F33D    39                 RTS                      ;RETURN TO CALLER
F33E    8601      MONFIX   LDAA     ¬$1
F340    B74006             STAA     PDAY            ;RESET DAY TO 1
F343    B64005             LDAA     PMONTH
F346    4C                 INCA
F347    B74005             STAA     PMONTH          ;INCREMENT THE MONTH
F34A    39                 RTS                      ;RETURN TO CALLER
                  ********************
                  **LEFT SHIFT TEMP1,2,3,4.
F34B    78401F    TEMPSH   ASL      TEMP4
F34E    79401E             ROL      TEMP3
F351    79401D             ROL      TEMP2
F354    79401C             ROL      TEMP1
F357    39                 RTS
                  ********************
                  **TURNS ALL POWWER OFF.
F358    B6E000    POWOFF   LDAA     CLOCKA
```

```
F35B    8A20               CFAA    ¬$$20
F35D    B7E000             STAA    CLOCKA
F360    8A18               CFAA    ¬$$18
F362    84DF               ANDA    ¬$$DF
F364    B7E000             STAA    CLOCKA
F367    39                 FTS
                  ********************
                  *↑TURNS CFF ↑-15(S),↑5(S).
F368    B6E000    WRMPOW   LCAA    CLOCKA
F36B    8A10               CFAA    ¬$$10
F36D    B7E000             STAA    CLOCKA
F370    8A28               CRAA    ¬$$28
F372    84EF               ANDA    ¬$$EF
F374    B7E000             STAA    CLOCKA
F377    39                 FTS
                  ********************
                  *↑THIS SLBROUTINE MOVES WHAT IS IN
                  *PMONTH-FTIMEL INTO MONTH-TIMEL.
F378    CE4001    UNSTOF   LCX     ¬$MONTH          ;↑I↑R=LOCATION OF MONTH
F37B    A604      UNROCK   LCAA    04,X             ;↑A=↑I↑R+4
F37D    A700               STAA    00,X             ;STORE IT IN ↑I↑R
F37F    08                 INX                      ;INCREMENT ↑I↑R
F380    8C4005             CFX     ¬$PMONTH          ;IF EQUAL WE ARE DONE
F383    26F6               BNE     UNROCK           ;LOOP BACK IF NOT
F385    39                 FTS
                  *********************
                  *[↑A↑B]↑[↑A↑B]-(WARM-UP TIME)
F386    F0401B    SUB*TI   SLBB    WTIME
F389    8200               SBCA    ¬$$00
F38B    39                 FTS
                  **********************
                  *↑INPUTS CATE OF THE FORM ↑6↑6/↑6↑6.
                  *↑STORES INPUT IN MONTH AND DAY.
F38C    36        DATE     FSHA
F38D    37                 FSHB
F38E    BDF15A             JSR     TWODIG           ;INPUT TWO DIGITS
F391    B74001             STAA    MONTH            ;STORE THEM IN MONTH
F394    3E                 WAI                      ;WAIT FOR NEXT CHAR
F395    BDF15A             JSR     TWODIG           ;INPUT TWO DIGITS
F398    B74002             STAA    DAY              ;STORE IN DAY
F39B    33                 FLLB
F39C    32                 FLLA
F39D    39                 FTS
                  ********************
                  *↑INPUTS TIME OF THE FORM ↑6↑6↑:↑6↑6.
                  *↑STORES INPUT IN TIMEH AND TIMEL.
F39E    36        TIME     FSHA
F39F    37                 FSHB
F3A0    BDF15A             JSR     TWODIG           ;INPUT TWO DIGITS
F3A3    5F                 CLRB                     ;CLEAR ↑B
F3A4    48                 ASLA                     ;≠2
F3A5    48                 ASLA                     ;≠4
F3A6    B74004             STAA    TIMEL            ;STORE TEMPORARILY HERE
F3A9    48                 ASLA                     ;≠8
F3AA    48                 ASLA
F3AB    59                 FCLB                     ;≠16
F3AC    48                 ASLA
```

```
F3AD   59                ROLB                         ;*32
F3AE   48                ASLA
F3AF   59                ROLB                         ;*64
F3B0   B04004            SUBA     TIMEL               ;[↑B↑A]=60*(2 DIGITS)
F3B3   C200              SBCB     #$00
F3B5   B74004            STAA     TIMEL
F3B8   F74003            STAB     TIMEH               ;STORE THE TIME IN TIMEH+L
F3BB   3E                WAI                          ;WAIT FOR ANOTHER INPUT
F3BC   BDF15A            JSR      TWODIG              ;GET TO MINUTE DIGITS OF TIME
F3BF   5F                CLRB                         ;CLEAR ↑B
F3C0   BB4004            ADDA     TIMEL
F3C3   F94003            ADCB     TIMEH               ;ADD FIRST TWO DIGITS IN MINUTES
F3C6   B74004            STAA     TIMEL
F3C9   F74003            STAB     TIMEH               ;STORE THE TIME
F3CC   33                PULB
F3CD   32                PULA
F3CE   39                RTS
              ************************
              **↑COMPARES [PDAY ↑WARMUP TIME] TO
              *SDAY RETURNS 0 IN ↑A IF NOT THE
              *SAME,1 IF THE SAME.
F3CF   BDF592 PSCMPW     JSR      STIME               ;SAVE PMONTH-LMINL
F3D2   B6401B            LDAA     WTIME               ;↑A=WARMUP TIME
F3D5   B7400A            STAA     LMINL
F3D8   7F4009            CLR      LMINH               ;SET LAST CLOCK TIME TO WARMUP TIME
F3DB   BDF2E1            JSR      UPTIME              ;ADD THE WARMUP TIME TO THE PRESENT TIME
F3DE   BDF261            JSR      PSCOMP              ;NOW COMPARE PRESENT TO STARTING TIME
F3E1   BDF583            JSR      RTIME               ;RESTORE PMONTH-LMINL
F3E4   4D                TSTA                         ;TEST ↑A
F3E5   39                RTS                          ;RETURN TO CALLER
              ************************
              **↑THE CONTENTS OF ↑A ARE CONVERTED
              *TO TWO ↑B↑C↑D DIGITS.
F3E6   B7401D BCD        STAA     TEMP2
F3E9   7F401C            CLR      TEMP1
F3EC   FE401C            LDX      TEMP1               ;↑I↑R=↑A
F3EF   4F                CLRA                         ;CLEAR ↑A
F3F0   8C0000 BCDLOP     CPX      #$00                 ;IF ZERO THEN CONVERSION DONE
F3F3   2706              BEQ      BCDDON              ;IF EQUAL BRANCH TO BCDDON
F3F5   09                DEX                          ;DECREMENT ↑I↑R
F3F6   8B01              ADDA     #$01                 ;INCREMENT ↑A
F3F8   19                DAA                          ;BCD ADJUST ↑A
F3F9   20F5              BRA      BCDLOP              ;BRANCH BACK FOR MORE
F3FB   39     BCDDON     RTS                          ;RETURN TO CALLER
              ***********************
              **↑PRINTS THE USER↑7S OPTIONS.
F3FC   BDF7C0 DISPCH     JSR      ALERT               ;PRINTS WARMUP POWER (↑O↑N,↑O↑F↑F).
F3FF   CEF406            LDX      #$DSPEC
F402   BDF717            JSR      PRINT               ;PRINT THE OPTIONS
F405   39                RTS
              **********************
F406   412920 DSPEC      FCC      ↑7A) INITIALIZING↑7
F409   494E49
F40C   544941
F40F   4C495A
F412   494E47
F415   205041            FCC      ↑7 PARAMETERS ↑7
```

```
F418   52414C
F41B   455445
F41E   525320
F421   0D0A7F           FCB     $0D,$0A,$7F,$7F
F424   7F
F425   422920           FCC     ↑7B) CURRENT SENSOR ↑7
F428   435552
F42B   52454E
F42E   542053
F431   454E53
F434   4F5220
F437   524541           FCC     ↑7READINGS↑7
F43A   44494E
F43D   4753
F43F   0D0A7F           FCB     $0D,$0A,$7F,$7F
F442   7F
F443   432920           FCC     ↑7C) SEQUENTIAL READINGS↑7
F446   534551
F449   55454E
F44C   544941
F44F   4C2052
F452   454144
F455   494E47
F458   53
F459   0D0A7F           FCB     $0D,$0A,$7F,$7F
F45C   7F
F45D   442920           FCC     ↑7D) INITIALIZATION↑7
F460   494E49
F463   544941
F466   4C495A
F469   415449
F46C   4F4E
F46E   0D0A7F           FCB     $0D,$0A,$7F,$7F
F471   7F
F472   452920           FCC     ↑7E) DUMP↑7
F475   44554D
F478   50
F479   0D0A7F           FCB     $0D,$0A,$7F,$7F,$20,$20,$20,$20,$20
F47C   7F2020
F47F   202020
F482   28434F           FCC     /(CONTROL ↑7S↑7 STOPS PRINTOUT)/
F485   4E5452
F488   4F4C20
F48B   275327
F48E   205354
F491   4F5053
F494   205052
F497   494E54
F49A   4F5554
F49D   29
F49E   0D0A7F           FCB     $0D,$0A,$7F,$7F,00
F4A1   7F00
                 ********************
                 **THIS SUBROUTINE DISABLES THE MEMORY.
F4A3   36        DISMEM PSHA
F4A4   863F             LDAA    #$3F
F4A6   B4E000           ANDA    CLOCKA
```

```
F4A9    B7E00D         STAA    CLOCKA
F4AC    32             PLLA
F4AD    39             FTS
               ********************
               *↑SHIFT LEFT [↑A↑B].
F4AE    58     SHFROL  ASLB
F4AF    49             FCLA
F4B0    39             FTS
               ********************
               *↑COMPARES PDATE TO FDATE AND
               *RETURNS A ZERO IF NOT THE
               *SAME.
F4B1    37     PFCOMP  FSHB
F4B2    F64011         LCAB    SMONTH
F4B5    F7401C         STAB    TEMP1          ;SAVE SMONTH
F4B8    F64012         LCAB    SDAY
F4BB    F7401D         STAB    TEMP2          ;SAVE SDAY
F4BE    B64017         LCAA    FMONTH
F4C1    B74011         STAA    SMONTH         ;PUT FMONTH IN SMONTH
F4C4    B64018         LCAA    FDAY
F4C7    B74012         STAA    SDAY           ;PUT FDAY IN SDAY
F4CA    BDF261         JSR     PSCOMP         ;COMPARE SDATE TO PDATE
F4CD    F6401C         LCAB    TEMP1
F4D0    F74011         STAB    SMONTH         ;RESTORE SMONTH
F4D3    F6401D         LCAB    TEMP2
F4D6    F74012         STAB    SDAY           ;RESTORE SDAY
F4D9    33             PLLB
F4DA    4D             TSTA                   ;TEST ↑A
F4DB    39             FTS                    ;RETURN TO CALLER
               ******************
               *↑THIS SLBROUTINE MOVES SMONTH-
               *LFREG INTO LOCATIONS PMONTH-
               *LMINL.
F4DC    CE4005 STOPS   LCX     ¬$PMONTH
F4DF    A60C   ROCK    LCAA    12,X           ;LOAD THE NEW WORD
F4E1    A700           STAA    $00,X          ;WRITE OVER OLD WORK
F4E3    08             INX                    ;INCREMENT ↑I↑R
F4E4    8C4008         CFX     ¬$LMINL+1       ;COMPARE TO LMINL+1
F4E7    26F6           BNE     ROCK           ;STOP IF EQUAL
F4E9    39             FTS                    ;RETURN TO CALLER
               ***************************
               *↑THIS SLBROUTINE INCREMENTS THE ↑I↑R
               *EQUAL TC THE NUMBER OF MEMORY
               *LOCATIONS NEEDED TO STORE EACH
               *SET OF READINGS.
F4EA    B6402E IRLOOP  LCAA    INCIR          ;↑A=¬$ OF INCREMENTS
F4ED    08     IRLOP   INX                    ;INCREMENT ↑I↑R
F4EE    4A             CECA                   ;DECREMENT ↑A
F4EF    26FC           BNE     IRLOP          ;IF ↑A EQUALS 0 QUIT
F4F1    39             FTS                    ;RETURN TO CALLER
               **********************
               *↑THIS SLBROUTINE TAKES THE DATE
               *AND TIME THAT ARE IN MONTH,DAY,
               *TIMEH, AND TIMEL AND COMES BACK
               *WITH THE ADDRESS OF THE FIRST
               *MEMORY LCCATION THAT HOLDS THAT
               *READING IN THE ↑I↑R.  ↑IF THAT
```

```
                    *READING WAS NEVER TAKEN THEN
                    **↑I↑R=0000.
F4F2   BDF592   RTAKEN  JSR    STIME        ;SAVE PMONTH-LMINL
F4F5   BDF4DC           JSR    STOPS        ;MOVES SMONTH-LFREQ TO MONTH-LMINL
F4F8   CE4040           LDX    ¬$STARTM      ;↑I↑R=STARTING LOCATION OF DATA
F4FB   BC4025   LOCLOP  CPX    MDATAH       ;IS ↑I↑R PAST CURRENT READINGS↑/
F4FE   2713             BEQ    OVERF        ;IS YES THEN READING NOT TAKEN
F500   BDF520           JSR    PINCMP       ;COMPARES MONTH-TIMEL TO PMONTH-LMINL
F503   2611             BNE    RETUNE       ;IF NOT 0, THEN LOCATION FOUND
F505   01               NOP
F506   01               NOP
F507   01               NOP
F508   BDF2E1   RJMP    JSR    UPTIME       ;IF 0, THEN INCREMENT THE TIME
F50B   01               NOP
F50C   01               NOP
F50D   01               NOP
                                            ;AND TRY AGAIN
F50E   BDF4EA           JSR    IRLOOP       ;INCREMENT ↑I↑R THE CORRECT AMOUNT
F511   20E8             BRA    LOCLOP       ;BRANCH BACK FOR TEST
F513   CE0000   OVERF   LDX    ¬$$0000       ;READING NOT TAKEN
F516   FF4021   RETUNE  STX    TEMP12       ;STORE TEMP12
F519   BDF5B3           JSR    RTIME        ;RESTORE PMONTH-LMINL
F51C   FE4021           LDX    TEMP12       ;RESORE ↑I↑R
F51F   39               RTS                 ;RETURN TO CALLER
                    ********************
                    ********************
                    **THIS SUBROUTINE COMPARES MONTH,
                    *DAY,TIMEH,TIMEL TO PMONTH,
                    *PDAY,PTIMEH,PTIMEL RESPECTIVELY.
                    **IF MONTH...TIMEL OCCURED BEFORE
                    *PMONTH...PTIMEL, THEN A ONE
                    *IS RETURNED IN ↑A OTHEWISE A
                    *ZERO IS RETURNED.
F520   FF4021   PINCMP  STX    TEMP12       ;SAVE ↑I↑R
F523   CE4001           LDX    ¬$MONTH       ;↑I↑R = ADDRESS OF MONTH
F526   8C4003   XLOOP   CPX    ¬$TIMEH       ;IS ↑I↑R=ADDRESS OF PMONTH
F529   2709             BEQ    ALEQUAL      ;NOW CHECK TIME
F52B   A600             LDAA   $00,X        ;ONE OF FIRST WORDS
F52D   A104             CMPA   $04,X        ;COMPARE TO ONE OF SECOND WORDS
F52F   221E             BHI    XOVER        ;IF FIRST TIME IS AFTER SECOND
                                            ;THEN THIS IS NOT TIME
F531   08               INX                 ;INCREMENT ↑I↑R
F532   20F2             BRA    XLOOP        ;BRANCH BACK TO TEST NEXT LOCATION
F534   F64003   ALEQUAL LDAB   TIMEH
F537   B64004           LDAA   TIMEL
F53A   B04008           SUBA   PTIMEL
F53D   F24007           SBCB   PTIMEH
F540   2D09             BLT    EQUAL
F542   2702             BEQ    SECCHEK
F544   2009             BRA    XOVER
F546   4D       SECCHEK TSTA
F547   2702             BEQ    EQUAL
F549   2004             BRA    XOVER
F54B   8601     EQUAL   LDAA   ¬$$01          ;TIME FOUND LOAD 1
F54D   2001             BRA    XDON         ;GOTO DONE
F54F   4F       XOVER   CLRA                ;NOT FOUND LOAD 0
F550   FE4021   XDON    LDX    TEMP12       ;RESTORE ↑I↑R
```

```
F553   4D                TSTA                      ;TEST ↑A
F554   39                RTS                       ;RETURN TO CALLER
              *******************
              *↑THIS SUBROUTINE PRINTS A
              *LINE FEED (↑L↑F) AND A CARRAGE
              *RETURN (↑C↑R).  ↑RUBOUTS ARE
              *INSERTED INCASE A TELETYPE
              *IS USED.
F555   36     LFCR       PSHA
F556   860D              LDAA     #$0D
F558   BDF2A0            JSR      OUTA
F55B   860A              LDAA     #$0A
F55D   BDF2A0            JSR      OUTA
F560   867F              LDAA     #$7F
F562   BDF2A0            JSR      OUTA
F565   BDF2A0            JSR      OUTA
F568   32                PULA
F569   39                RTS
              *********************
              ***********************
              *↑THIS SUBROUTINE SHIFTS A
              *16 BIT WORD 4 BITS TO THE LEFT.
              *↑THE 16 BIT WORD ARE 2 MEMORY
              *LOCATIONS, THE FIRST IS POINTED
              *TO BY THE ↑I↑R.
F56A   4F     LSHFT4     CLRA                      ;CLEAR ↑A
F56B   6801   AROUND     ASL      $01,X            ;SHIFT LOW ORDER WORD
F56D   6900              ROL      $00,X            ;SHIFT HGH ORDER WORD
F56F   4C                INCA                      ;INCREMENT ↑A
F570   8104              CMPA     #$04              ;TEST FOR 4 SHIFTS
F572   26F7              BNE      AROUND           ;IF NOT DONE LOOP BACK
F574   39                RTS                       ;RETURN TO CALLER
              **********************
              *↑THIS SUBROUTINE SHIFTS A 16 BIT
              *WORD 1 BIT TO THE RIGHT.  ↑THE WORD
              *IS POINTED TO BY THE ↑I↑R.
F575   6400   SHIFTR     LSR      00,X             ;SHIFT THE HIGH ORDER WORD
F577   6601              ROR      01,X             ;SHIFT THE LOW ORDER WORD
F579   39                RTS                       ;RETURN TO CALLER
              **********************
              *↑THIS SUBROUTINE PRINTS OUT ONE
              *BYTE.  ↑THE BYTE IS IN ↑A.
F57A   37     BYTOUT     PSHB                      ;SAVE ↑B
F57B   16                TAB                       ;↑B=↑A
F57C   44                LSRA
F57D   44                LSRA
F57E   44                LSRA
F57F   44                LSRA                      ;SHIFT ↑A RIGHT 4 BITS
F580   C40F              ANDB     #$0F              ;MASK OUT UPPER 4 BITS
F582   37                PSHB                      ;SAVE ↑B
F583   BDF68D            JSR      HEXASC           ;CONVERT TO HEX
F586   BDF2A0            JSR      OUTA             ;PRINT
F589   32                PULA                      ;GET LOWER 4 BITS
F58A   BDF68D            JSR      HEXASC           ;CONVERT TO HEX
F58D   BDF2A0            JSR      OUTA             ;PRINT ↑A
F590   33                PULB                      ;RESORE ↑B
F591   39                RTS                       ;RETURN TO CALLER
```

```
                  ********************
                  *↑TLIS SLEROUTINE SAVES PMONTH-
                  *LMINL IN TEMPORARY LOCAIONS.
F592    36        STIME    FSHA                   ;SAVE ↑A
F593    37                 FSHB                   ;SAVE ↑B
F594    BDF616             JSR      SAT5          ;CHECK IF AN STIME
                                                  ;HAS BEEN DONE
                                                  ;WITHOUT AN RTIME FOLLOWING IT
F597    2602               BNE      CSTIME        ;IF IT HASN↑7T THEN BRANCH
F599    2015               BRA  -   SDON          ;ILLEGAL  REQUEST LEAVE SUBROUTINE
F59B    CE4005    CSTIME   LCX      ¬$PMONTH       ;↑I↑R=ADDRESS OF PMONTH
F59E    A600      NOCH1    LCAA     $00,X         ;↑A=VALUE TO BE MOVED
F5A0    A706               STAA     $06,X         ;STORE VALUE IN A TEMP. LOC.
F5A2    08                 INX                    ;INCREMENT ↑I↑R
F5A3    8C4006             CPX      ¬$TEMP6        ;ARE WE DONE↑/
F5A6    26F6               BNE      NOCH1         ;IF NO LOOP BACK
F5A8    8620               LCAA     ¬$$20
F5AA    BA4000             CRAA     STATUS
F5AD    B74000             STAA     STATUS        ;SET STATUS BIT SHOWING AN
                                                  ;STIME HAS BEEN DONE
F5B0    33        SDON     PLLB                   ;RESTORE ↑B
F5B1    32                 PLLA                   ;RESTORE ↑A
F5B2    39                 RTS                    ;RETURN TO CALLER
                  ********************
                  *↑THIS SLEROUTINE RESTORES THE VALUES
                  *OF PMONTH-LMINL.  (↑NOTE↑; ↑IF AN STIME
                  *HAS NOT BEEN DONE THIS SUBROUTINE
                  *WILL DO NOTHING.)
F5B3    36        RTIME    FSHA                   ;SAVE ↑A
F5B4    37                 FSHB                   ;SAVE ↑B
F5B5    BDF616             JSR      SAT5          ;CHECK FOR STIME DONE
F5B8    2702               BEQ      CRTIME        ;IF AN STIME HAS BEEN DONE BRANCH
F5BA    2015               BRA      RDON          ;STIME NOT DONE EXIT ROUTINE
F5BC    CE4005    CRTIME   LCX      ¬$PMONTH       ;↑I↑R=PMONTH
F5BF    A606      NOCH2    LCAA     $06,X         ;LOAD VALUE
F5C1    A700               STAA     $00,X         ;RESORE THE VALUE
F5C3    08                 INX                    ;INCREMENT ↑I↑R
F5C4    8C4006             CPX      ¬$TEMP6        ;ARE WE DONE↑/
F5C7    26F6               BNE      NOCH2         ;IF NO, THEN LOOP BACK
F5C9    86DF               LCAA     ¬$$DF
F5CB    B44000             ANDA     STATUS
F5CE    B74000             STAA     STATUS        ;RESET STATUS BIT
F5D1    33        RDON     PLLB                   ;RESTORE ↑B
F5D2    32                 PLLA                   ;RESTORE ↑A
F5D3    39                 RTS                    ;RETURN TO CALLER
                  ********************
                  *↑THIS SLEROUTINE TAKES WHAT IS
                  *IN ↑B ANC PRINTS IT.  ↑THEN IT
                  *PRINTS WHATS IN TEMP5 (USUALLY
                  *A ↑7↑;↑7 OR A ↑7/↑7).  ↑AND FOLLOWS
                  *THAT BY THE CHARACTER IN ↑A.
F5D4    36        VAROUT   FSHA                   ;SAVE ↑A
F5D5    17                 TBA                    ;↑A=↑B
F5D6    BDF3E6             JSR      BCD           ;CONVERT TO BCD
F5D9    BDF57A             JSR      BYTOUT        ;PRINT THE TWO CHARS
F5DC    B64020             LCAA     TEMP5
F5DF    BDF2A0             JSR      OUTA          ;PRINT WHAT IS IN TEMP5
```

```
F5E2   32                PLLA                       ;RESTORE ↑A
F5E3   BDF3E6            JSR     BCD                ;CONVERT TO BCD
F5E6   BDF57A            JSR     BYTOUT             ;PRINT BOTH CHARS
F5E9   39                FTS                        ;RETURN TO CALLER
              *********************
              **THIS SLBROUTINE PRINTS THE DATE.
              **THE MONTH MUST BE IN ↑B AND THE
              *DAY MUST BE IN ↑A.
F5EA   36     PDATE      FSHA                       ;SAVE ↑A
F5EB   862F              LCAA    ¬$$2F
F5ED   B74020            STAA    TEMP5              ;TEMP5=↑7/↑7
F5F0   32                PLLA                       ;RESTORE ↑A
F5F1   BDF5D4            JSR     VAROUT             ;VAROUT PRINTS OUT DATE
F5F4   39                FTS                        ;RETURN TO CALLER
              *****************
              **THIS SLBROUTINE PRINTS OUT
              *FIVE SPACES.
F5F5   36     PSPACE     FSHA                       ;SAVE ↑A
F5F6   37                FSHB                       ;SAVE ↑B
F5F7   CEF000            LCX     ¬$SPACES            ;LOAD ↑I↑R WITH ADDRESS
F5FA   BDF717            JSR     PRINT              ;PRINT ↑7     ↑7
F5FD   33                PLLB                       ;RESTORE ↑B
F5FE   32                PLLA                       ;RESTORE ↑A
F5FF   39                FTS                        ;RETURN TO CALLER
              ********************
              **THIS SLBROUTINE PRINTS ONE SPACE.
F600   36     PSP        FSHA                       ;SAVE ↑A
F601   37                FSHB                       ;SAVE ↑B
F602   8620              LCAA    ¬$$20
F604   BDF2A0            JSR     OUTA               ;PRINT ↑7 ↑7
F607   33                PLLB                       ;RESTORE ↑B
F608   32                PLLA                       ;RESTORE ↑A
F609   39                FTS                        ;RETURN TO CALLER
              *******************
              **THIS SLBROUTINE CLEARS THE STACK
              *IN EXACTLY THE SAME MANNER THAT
              *A ↑R↑T↑I INSTRUCTION WOULD, BUT WITH-
              *OUT THE RETURN JUMP.  (↑A ↑R↑T↑I IS
              *A RETURN FROM AN INTERRUPT.)
F60A   32     FAKERI     PLLA                       ;SAVE ↑A (IN ↑A)
F60B   33                PLLB                       ;SAVE ↑B (IN ↑B)
F60C   31                INS
F60D   31                INS
F60E   31                INS
F60F   31                INS
F610   31                INS
F611   31                INS
F612   31                INS                        ;INCREMENT THE STACK POINTER SEVEN
                                                    ;TIMES JUST LIKE AN ↑R↑T↑I
F613   37                FSHB                       ;PUSH ↑B BACK ON STACK
F614   36                FSHA                       ;PUSH ↑A BACK ONTO THE STACK
F615   39                FTS                        ;RETURN TO CALLER
              *******************
              **THIS SLBROUTINE CHECKS THE
              *STATUS BIT THAT IS SET AND
              *CLEARED BY RTIME AND STIME.
F616   8620   SATS       LCAA    ¬$$20               ;PUT A ONE IN THE CORRECT BIT
```

C-2

```
F618    B44000          ANDA    STATUS          ;↑A↑N↑D THE STATUS WORD
F61B    8020            SUBA    ¬$$20            ;SUBTRACT 20(HEX) TO TEST BIT
F61D    39              RTS                     ;RETURN TO CALLER
                ********************
                *↑THIS SUBROUTINE INHIBITS
                *THE CLOCK BOARD FROM GEN-
                *ERATING AN ↑N↑M↑I (INTERRUPT).
F61E    36      OFFCLK  PSHA                    ;SAVE ↑A
F61F    8604            LDAA    ¬$$04            ;SET THE PROPER BIT
F621    BAE002          ORAA    CLOCKB          ;FORM WORD TO WRITE TO CLOCK
F624    B7E002          STAA    CLOCKB          ;WRITE WORD TO CLOCK
F627    32              PULA                    ;RESTORE ↑A
F628    39              RTS                     ;RETURN TO CALLER
                ********************
                *↑THIS SUBROUTINE ENABLES THE
                *MEMORY.  ↑THE TWO BITS THAT
                *CONTROL THIS ARE ON THE CLOCK
                *BOARD (PORT ↑B).
F629    36      ENMEM   PSHA                    ;SAVE ↑A
F62A    867F            LDAA    ¬$$7F            ;ZERO CORRECT BIT
F62C    B4E000          ANDA    CLOCKA          ;FORM WORD TO WRITE TO CLOCK
F62F    8A40            ORAA    ¬$$40            ;SET (1) CORRECT BIT
F631    B7E000          STAA    CLOCKA          ;WRITE WORD TO CLOCK
F634    32              PULA                    ;RESTORE ↑A
F635    39              RTS                     ;RETURN TO CALLER
                *********************
                *↑THIS SUBROUTINE ENABLES THE
                *INTERRUPT FROM THE CLOCK BOARD.
F636    B6E002  CLICLK  LDAA    CLOCKB
F639    84FB            ANDA    ¬$$FB            ;RESET BIT 2
F63B    B7E002          STAA    CLOCKB          ;WRITE TO CLOCK
F63E    39              RTS                     ;RETURN TO CALLER
                ************
                *↑THIS SUBROUTINE TAKES THE NUMBER
                *IN [↑A↑B] AND STUFFS IT INTO THE
                *TIMER ON THE CLOCK BOARD.  ↑THE
                *TIMER IS ONLY 12 BITS AND THE
                *4 HIGH ORDER BITS OF ↑A ARE
                *ASSUMED TO BE ZERO (DISCARDED).
                *(↑BIT 2 OF CLOCKA IS THE DATA IN
                *TO THE SHIFT REGISTER.  ↑BIT 1
                *IS THE CLOCK ON THE SHIFT REG-
                *ISTER. )
F63F    B74009  CLKSET  STAA    LMINH           ;STORE TIME IN LMINH
F642    F7400A          STAB    LMINL           ;STORE TIME IN LMINL
F645    BDF4AE          JSR     SHFROL
F648    BDF4AE          JSR     SHFROL
F64B    BDF4AE          JSR     SHFROL          ;SHIFT LEFT [↑A↑B] 3 TIMES
F64E    CE000C          LDX     ¬$12             ;↑I↑R=12
F651    BDF4AE  OVERHER JSR     SHFROL          ;SHIFT LEFT [↑A↑B]
F654    36              PSHA                    ;SAVE ↑A
F655    4D              TSTA                    ;TEST HIGH ORDER BIT
F656    2C14            BGE     BITZERO         ;IF HIGHEST BIT IS 0 BRANCH
F658    B6E000          LDAA    CLOCKA
F65B    8A04            ORAA    ¬$$04            ;FORM CLOCK WORD WITH DATA IN = 1
F65D    B7E000          STAA    CLOCKA          ;WRITE WORD TO CLOCK
F660    8A06            ORAA    ¬$$06
```

```
F662   B7E000          STAA    CLOCKA          ;MAKE SHIFT CLOCK GO HIGH
F665   84FC            ANDA    ¬$$FC
F667   B7E000          STAA    CLOCKA          ;MAKE SHIFT CLOCK GO LOW
F66A   2012            BFA     RIGHTH          ;CLEAN UP BEFORE LOOPING BACK
F66C   B6E000  BITZERO LCAA    CLOCKA          ;HIGHEST BIT IS ZERO
F66F   84F8            ANDA    ¬$$F8
F671   B7E000          STAA    CLOCKA          ;SET INPUT BIT TO ZERO
F674   8A02            CRAA    ¬$$02
F676   B7E000          STAA    CLOCKA          ;SET SHIFT CLOCK TO HIGH
F679   84F8            ANDA    ¬$$F8
F67B   B7E000          STAA    CLOCKA          ;RETURN SHIFT CLOCK LOW
F67E   32      RIGHTH  PLLA                    ;RESTORE ↑A
F67F   09              CEX                     ;DECREMENT ↑I↑R
F680   26CF            BNE     OVERHER         ;IF NOT ZERO LOOP BACK
F682   39              FTS                     ;RETURN TO CALLER
               **********************
               *↑THIS SLBROUTINE CONVERTS THE
               *ASCII CHARACTER IN ↑A INTO A
               *HEX NUMBER.  (↑IT IS ASSUMED
               *THAT THE CHARACTER IS A
               *HEX NUMBER IN ASCII, IE. 0-F.)
F683   8039    ASCHEX  SLBA    ¬$$39            ;SUBTRACT 39 (HEX)
F685  -2E03            BCT     BETA            ;BRANCH IF CHAR WAS ↑A-↑F
F687   8B09            ACDA    ¬$$09            ;ADD 9
F689   39              FTS                     ;RETURN TO CALLER
F68A   8B02    BETA    ACDA    ¬$$02            ;IS ↑A,↑B,↑C,↑D,↑E, OR ↑F ADD 2
F68C   39              FTS                     ;RETURN TO CALLER
               **********************
               *↑THIS SLBROUTINE CONVERTS A
               *HEX NUMBER INTO A ASCII CHAR.
F68D   8BF7    HEXASC  ACDA    ¬$$-9            ;SUBTRACT 9
F68F   2E03            BGT     ALPHA           ;BRANCH IF ↑A-↑F
F691   8B39            ACDA    ¬$$39            ;ADD 39 (HEX)
F693   39              FTS                     ;RETURN TO CALLER
F694   8B40    ALPHA   ACDA    ¬$$40            ;ADD 40 (HEX)
F696   39              FTS                     ;RETURN TO CALLER
               **********************
               *↑THIS SLBROUTINE INTIALIZES
               *ALL OF THE PIAS IN THE SYSTEM.
               *↑THERE ARE TWO PIAS↑; THE CLOCK
               *AND THE A/D PIA.
F697   7FE801  PIAS    CLR     ADPIAA+1
F69A   7FE803          CLR     ADPIAB+1
F69D   7FE001          CLR     CLOCKA+1
F6A0   7FE003          CLR     CLOCKB+1        ;CLEAR THE CONTROL REGISTER
F6A3   4F              CLRA
F6A4   B7E800          STAA    ADPIAA          ;CONFIGURE ALL BITS AS INPUT
F6A7   86E0            LCAA    ¬$$E0
F6A9   B7E802          STAA    ADPIAB          ;CONFIGURE BITS 5,6, AND 7 AS OUTPUT
                                               ;THE REST AS INPUT
F6AC   8604            LCAA    ¬$$04
F6AE   B7E801          STAA    ADPIAA+1        ;DISABLE INTERRUPTS AND
                                               ;SET LOCATION
                                               ;TO THE PERIPHERAL REGISTER
F6B1   B7E802          STAA    ADPIAB          ;DISABLE INTERRUPTS AND SET
                                               ;TO THE PERIPHERAL REGISTER
F6B4   4A              CECA
```

```
F6B5   B7D800            STAA   CNTRL          ;RESET THE ↑A↑C↑I↑A
F6B8   8689              LDAA   ¬$$89
F6BA   B7D800            STAA   CNTRL          ;SET CLOCK RATE, WORD LENGTH,
F6BD   86FF              LDAA   ¬$$FF           ;AND INTERRUPT
F6BF   B7E000            STAA   CLOCKA         ;CONFIGURE ALL BITS AS OUTPUT
F6C2   8605              LDAA   ¬$$05
F6C4   B7E001            STAA   CLOCKA+1       ;SET INTERRUPT TO THE NEGATIVE EDGE
                                               ;AND CHANGE ACCESS TO PERIPHHERAL
F6C7   86F7              LDAA   ¬$$F7
F6C9   B7E002            STAA   CLOCKB         ;SET BIT 3 AS INPUT, THE REST AS OUTPUT
F6CC   8604              LDAA   ¬$$04
F6CE   B7E003            STAA   CLOCKB+1       ;CHANGE ACCESS TO THE PERIPHERAL
                                               ;AND DISABLE ALL INTERRUPTS
F6D1   8630              LDAA   ¬$$30
F6D3   B7E000            STAA   CLOCKA         ;DISABLE MEMORY WITH THE NORMAL BITS
F6D6   B6E000            LDAA   CLOCKA         ;CLEAR ANY INTERRUPT ON PORT ↑A
F6D9   39                RTS
                 **********************
                 *↑THIS SUBROUTINE CONVERTS AN ANALOG VOLTAGE
                 *ON PORT SPECIFIED BY SENNUM TO A 12 BIT
                 *DIGITAL NUMBER AND STORES IT IN CONVH,CONVL.
F6DA   36        ADCVAL  PSHA
F6DB   37                PSHB
F6DC   B6402C            LDAA   SENNUM
F6DF   48                ASLA
F6E0   48                ASLA
F6E1   48                ASLA
F6E2   48                ASLA
F6E3   48                ASLA
F6E4   B7E802            STAA   ADPIAB         ;SELECTS PORT
F6E7   BDF9E5            JSR    MDELAY         ;ALLOWS MULTIPLEXER TO SETTLE
F6EA   8620              LDAA   ¬$$20
F6EC   BAE002            ORAA   CLOCKB
F6EF   B7E002            STAA   CLOCKB         ;INITIATES ↑A/↑D CONVERSION
F6F2   860F              LDAA   ¬$$DF
F6F4   B4E002            ANDA   CLOCKB
F6F7   B7E002            STAA   CLOCKB
F6FA   BDF9E5            JSR    MDELAY         ;ALLOWS ↑A/↑D TIME TO START CONVERSON
F6FD   B6E802    READY   LDAA   ADPIAB
F700   8410              ANDA   ¬$$10
F702   26F9              BNE    READY          ;CHECKS STATUS OF ↑A/↑D CONVERSION
F704   B6E802            LDAA   ADPIAB         ;TAKES COMPLEMENTARY BINARY OUTPUT,
                                               ;CONVERTS TO SIGNED BINARY AND
                                               ;STORES IN CONVH,CONVL
F707   8AF0              ORAA   ¬$$F0
F709   43                COMA
F70A   B74023            STAA   CONVH
F70D   B6E800            LDAA   ADPIAA
F710   43                COMA
F711   B74024            STAA   CONVL
F714   33                PULB
F715   32                PULA
F716   39                RTS
                 *******************
                 *↑THIS SUBROUTINE PRINTS OUT
                 *CHARACTER STRINGS STORED IN
                 *THE PROGRAM.  ↑THE STARTING
```

```
                *ADDRESS OF THE CHARACTER
                *STRING MUST BE IN THE ↑I↑R
                *WHEN THE SUBROUTINE IS CALLED.
                *↑THE SUBROUTINE THEN PRINTS
                *OUT CHARACTER FROM CONSEC-
                *UTIVE MEMORY LOCATIONS UNTIL
                *IT READS A 00.
F717   36       PRINT  PSHA                  ;SAVE ↑A
F718   A600     BACKW  LDAA   $00,X          ;MOVE CHARACTER INTO ↑A
F71A   8100            CMPA   ¬$$00          ;IS IT A ZERO↑/
F71C   2602            BNE    HERE           ;IF IT ISN↑7T THEN BRANCH TO HERE
F71E   32              PULA                  ;RESTORE ↑A
F71F   39              RTS                   ;RETURN TO CALLER
F720   BDF2A0   HERE   JSR    OUTA           ;PRINT THE CHARACTER
F723   08              INX                   ;INCREMENT THE ↑I↑R
F724   20F2            BRA    BACKW          ;LOOP BACK FOR MORE CHARACTERS
                **********************
                *↑THIS SUBROUTINE PRINTS OUT
                *A TIME.  ↑THE TIME TO BE PRINTED
                *MUST BE IN MINUTES AND MUST
                *BE STORED IN [↑B↑A].
F726   CE0000   PTIME  LDX    ¬$$00          ;CLEAR ↑I↑R
F729   08       PLOOP  INX                   ;INCREMENT ↑I↑R (COUNT HOURS)
F72A   803C            SUBA   ¬$60           ;SUBTRACT SIXTY MINUTES
F72C   C200            SBCB   ¬$$00          ;SUBTRACT CARRY
F72E   2CF9            BGE    PLOOP          ;IF NOT NEGATIVE LOOP BACK
F730   8B3C            ADDA   ¬$60           ;MAKE TIME POSITIVE AGAIN
F732   09              DEX                   ;ADJUST ↑I↑R
F733   FF4021          STX    TEMP12         ;SAVE IN A TEMPORARY LOC.
F736   F64022          LDAB   TEMP13         ;WRITE ¬$ OF HOURS INTO ↑B
F739   36              PSHA                  ;SAVE ↑A
F73A   863A            LDAA   ¬$$3A
F73C   B74020          STAA   TEMP5          ;TEMP5=↑7↑;↑7
F73F   32              PULA                  ;RESTORE ↑A
F740   BDF5D4          JSR    VAROUT         ;PRINT THE TIME
F743   39              RTS                   ;RETURN TO CALLER
                *********************
                *↑MULTIPLIES SUCCESIVE MEMORY LOCATIONS SPECIFIED
                *BE INDEX REGISTER BY 2.44140625 MAINTAINING
                *THE RUNNING SUM IN ↑A AND ↑B (CONTENTS OF THE
                *MEMORY LOCATIONS ARE DESTROYED).
F744   A601     MULT1  LDAA   $01,X
F746   E600            LDAB   $00,X
F748   48              ASLA
F749   59              ROLB
F74A   BDF575          JSR    SHIFTR
F74D   BDF763          JSR    SHIFAD
F750   BDF763          JSR    SHIFAD
F753   BDF763          JSR    SHIFAD
F756   BDF575          JSR    SHIFTR
F759   BDF575          JSR    SHIFTR
F75C   BDF575          JSR    SHIFTR
F75F   BDF763          JSR    SHIFAD
F762   39              RTS
                **********************
                *↑THIS ROUTINE RIGHT SHIFTS THE 16
                *BIT WORD IN SUCCESIVE MEMORY LOCATIONS SPEC-
```

```
                    *IFIED BY THE INDEX REGISTER, THEN ADDS THEM
                    *TO CONTENTS OF ↑A,↑B (WITH ROUNDING).
F763    6400        SHIFAD  LSR     00,X
F765    6601                FCR     01,X
F767    A901                ACCA    01,X
F769    E900                ACCB    00,X
F76B    39                  FTS
                    ********************
                    *↑THIS SUBROUTINE CONVERTS THE BINARY NUMBER
                    *IN CONVH,CONVL TO ↑B↑C↑D AND RETURNS IT
                    *TO CONVH, CONVL.
F76C    FE4023      DECADJ  LDX     CONVH
F76F    4F                  CLRA
F770    5F                  CLRB
F771    8C0000      DONYET  CFX     ¬$$0000             ;CHECK IF ↑I↑R=0000
F774    2711                BEQ     FINISH
F776    09                  CEX                         ;DECREMENT ↑I↑R
F777    8B01                ACDA    ¬$$01                ;ADD ONE TO ↑A
F779    19                  CAA                         ;CONVERT ↑A TO A DECIMAL NUMBER
F77A    C900                ACCB    ¬$$00                ;ADD CARRY BIT TO ↑B
F77C    B74023              STAA    CONVH
F77F    17                  TBA
F780    19                  CAA                         ;CONVERT ↑B TO A DECIMAL NUMBER
F781    16                  TAB
F782    B64023              LCAA    CONVH
F785    20EA                BRA     DONYET
F787    B74024      FINISH  STAA    CONVL               ;STORE ↑B↑C↑D RESULT IN CONVL,CONVH
F78A    F74023              STAB    CONVH
F78D    39                  FTS
                    ********************
F78E    484F57      HOW     FCC     ↑7HOW MANY↑/↑7
F791    204D41
F794    4E593F
F797    00                  FCB     00
F798    285858      THREE   FCC     ↑7(XXX) ↑7
F79B    582920
F79E    00                  FCB     00
F79F    4F4E2E      ON      FCC     ↑7ON.↑7
F7A2    00                  FCB     00
F7A3    4F4646      OFF     FCC     ↑7OFF.↑7
F7A6    2E
F7A7    00                  FCB     00
F7A8    574152      POWIS   FCC     ↑7WARM-UP POWER IS ↑7
F7AB    4D2D55
F7AE    502050
F7B1    4F5745
F7B4    522049
F7B7    5320
F7B9    00                  FCB     00
F7BA    2A4445      DEL     FCC     ↑7*DEL*↑7
F7BD    4C2A
F7BF    00                  FCB     $00
                    ************************
                    *↑THIS SUBROUTINE PRINTS OUT THE
                    *STATE OF THE WARMUP POWER (↑O↑N,
                    *↑O↑F↑F).
F7C0    CEF7A8      ALERT   LCX     ¬$POWIS
```

```
F7C3   BDF717           JSR    PRINT       ;PRINT ↑7↑WARM-UP POWER IS↑7
F7C6   BDF629           JSR    ENMEM       ;ENABLE MEMORY
F7C9   B64000           LDAA   STATUS
F7CC   8402             ANDA   #$02         ;CHECK WARMUP POWER BIT
F7CE   270D             BEQ    WOFF        ;IF ZERO THEN IT IS OFF
F7D0   B6401B           LDAA   WTIME
F7D3   2708             BEQ    WOFF        ;IF WTIME=0 THEN THERE IS
                                           ;NO WARM-UP TIME
F7D5   CEF79F           LDX    #$ON
F7D8   BDF717           JSR    PRINT       ;PRINT ↑O↑N↑.
F7DB   2006             BRA    DALERT      ;BRANCH TO DALERT
F7DD   CEF7A3  WOFF     LDX    #$OFF
F7E0   BDF717           JSR    PRINT       ;PRINT ↑O↑F↑F↑.
F7E3   BDF555  DALERT   JSR    LFCR        ;SKIP A LINE
F7E6   BDF4A3           JSR    DISMEM      ;DISABLE MEMORY
F7E9   39               RTS                ;RETURN TO CALLER
                ********************
                        END

* THE BINARY IN IN PHYSICAL BLOCK 2
```

```
                  *
                  *
                  *
                           ORG     $F800
                  ****************************
                  **↑THIS SUBROUTINE APPLIES A LINEAR CONVERSION
                  *TO THE BINARY DATA IN CONVH,CONVL DEPENDING
                  *ON THE CONTENTS OF SENNUM↑;
                  *        SENNUM     OUTPUT RANGE
                  *          0        -30.00 - 70.00
                  *          1        -34.00 - 50.00
                  *          2        800.00 - 1100.00
                  *         >2         00.00 -  5.00
                  *↑THE OUTPUT IS PRINTED IN DECIMAL FORMAT.
F800    36        SENSCV   PSHA
F801    FF4026             STX     TEMPIR
F804    37                 PSHB
F805    CE4023             LDX     ¬$CONVH
F808    B6402C             LDAA    SENNUM
F80B    4D                 TSTA                       ;TEST FOR SENNUM=0, GOTO TEMPCV
F80C    2603               BNE     CHERE
F80E    7EF892             JMP     TEMPCV
F811    8101      CHERE    CMPA    ¬$$01               ;TEST FOR SENNUM=1, GOTO DEWPT
F813    2603               BNE     SKIPJMP
F815    7EF881             JMP     DEWPT
F818    8102      SKIPJMP  CMPA    ¬$$02               ;TEST FOR SENNUM=2, GOTO PRESUR
F81A    2603               BNE     HOP1
F81C    7EF8E4             JMP     PRESUR
F81F    BDF575    HOP1     JSR     SHIFTR             ;DIVIDE NUMBER BY 2
F822    A601               LDAA    $01,X
F824    E600               LDAB    $00,X
F826    BDF744             JSR     MULT1              ;MULTIPLY NUMBER BY 2.44140625
F829    A701               STAA    $01,X
F82B    E700               STAB    $00,X
F82D    BDF76C             JSR     DECADJ
F830    CE4023             LDX     ¬$CONVH
F833    A600               LDAA    $00,X
F835    84F0               ANDA    ¬$$F0
F837    44                 LSRA
F838    44                 LSRA
F839    44                 LSRA
F83A    44                 LSRA
F83B    8A30               ORAA    ¬$$30               ;CONVERT FIRST ↑B↑C↑D DIGIT
                                                      ;TO ASCII
F83D    BDF2A0             JSR     OUTA               ;PRINT THE DIGIT
F840    862E               LDAA    ¬$$2E
F842    BDF2A0             JSR     OUTA               ;PRINT THE DECIMAL POINT
F845    A600               LDAA    $00,X
F847    840F               ANDA    ¬$$0F
F849    8A30               ORAA    ¬$$30               ;CONVERT FIRST ↑B↑C↑D DIGIT
                                                      ;TO ASCII
F84B    BDF2A0             JSR     OUTA               ;PRINT THE DIGIT
F84E    A601               LDAA    $01,X
F850    BDF57A             JSR     BYTOUT             ;PRINT THE THIRD AND FOURTH DIGITS
F853    7EF889             JMP     MISSED
F856    BDF76C    TOGETR   JSR     DECADJ             ;CONVERT TO DECIMAL NUMBER
F859    B64023    JOINUP   LDAA    CONVH
```

```
F85C  4D               TSTA             ;CHECK FOR TWO LEADING ZEROS
F85D  2608             BNE    SECND     ;IF NOT, GO TO SECND
F85F  BDF600           JSR    PSP       ;PRINT A SPACE
F862  BDF600           JSR    PSP       ;PRINT ANOTHER SPACE
F865  2017             BRA    SKIP
F867  84F0    SECND    ANDA   #$F0       ;CHECK FOR ONE LEADING ZERO
F869  260D             BNE    PRTALL    ;IF NOT, GO TO PRTALL
F86B  BDF600           JSR    PSP       ;PRINT A SPACE
F86E  B64023           LDAA   CONVH
F871  8A30             ORAA   #$30       ;CONVERT SECOND DIGIT TO ASCII
F873  BDF2A0           JSR    OUTA      ;PRINT THE DIGIT
F876  2006             BRA    SKIP
F878  B64023  PRTALL   LDAA   CONVH
F87B  BDF57A           JSR    BYTOUT    ;PRINT BOTH LEADING DIGITS
F87E  862E    SKIP     LDAA   #$2E
F880  BDF2A0           JSR    OUTA      ;PRINT A DECIMAL POINT
F883  B64024           LDAA   CONVL
F886  BDF57A           JSR    BYTOUT    ;PRINT LAST TWO DIGITS
F889  BDF600  MISSED   JSR    PSP       ;PRINT A SPACE
F88C  FE4026           LDX    TEMPIR
F88F  33               PULB
F890  32               PULA
F891  39               RTS
F892  6F01    TEMPCV   CLR    $01,X
F894  BDF94A           JSR    RSHIFT4   ;SHIFT 8 BIT TEMP. WORD TO
                                        ;THE RIGHT 4 TIMES
F897  A601             LDAA   $01,X
F899  E600             LDAB   $00,X
F89B  BDF744           JSR    MULT1     ;MULT. BY 2.44140625
F89E  80B8             SUBA   #$B8       ;SUBTRACT 3000
F8A0  C20B             SBCB   #$0B       ;FORM THE RESULT
F8A2  A701             STAA   01,X
F8A4  E700             STAB   00,X
F8A6  2C03             BGE    NONEG     ;CHECK IF RESULT IS NEGATIVE
F8A8  7EF937           JMP    NEGNUM    ;IF SO, GO TO NEGNUM
F8AB  BDF600  NONEG    JSR    PSP       ;IF NOT, PRINT SPACE AND
                                        ;SHIFT TO THE RIGHT 4 TIMES
F8AE  7EF856           JMP    TOGETR
F8B1  6F01    DEWPT    CLR    $01,X
F8B3  BDF94A           JSR    RSHIFT4   ;SHIFT 8 BIT DEW PT. WORD TO
                                        ;THE RIGHT 4 TIMES
F8B6  A601             LDAA   $01,X
F8B8  E600             LDAB   $00,X
F8BA  48               ASLA
F8BB  59               ROLB             ;MULTIPLY DATA BY 2.05078125
F8BC  BDF575           JSR    SHIFTR
F8BF  BDF575           JSR    SHIFTR
F8C2  BDF575           JSR    SHIFTR
F8C5  BDF575           JSR    SHIFTR
F8C8  BDF763           JSR    SHIFAD
F8CB  BDF763           JSR    SHIFAD
F8CE  BDF575           JSR    SHIFTR
F8D1  BDF763           JSR    SHIFAD
F8D4  8048             SUBA   #$48       ;SUBTRACT 3400 FROM THE RESULT
F8D6  C20D             SBCB   #$0D
F8D8  A701             STAA   $01,X
F8DA  E700             STAB   $00,X
```

```
F8DC  2D03             BLT   HOPUP
F8DE  7EF8AB           JMP   NONEG      ;IF RESULT IS NOT NEGATIVE,
                                        ;BRANCH TO NONEG
F8E1  7EF937  HOPUP    JMP   NEGNUM     ;IF NOT JUMP TO NEGNUM
F8E4  A601    PRESUR   LDAA  $01,X
F8E6  E600             LDAB  $00,X
F8E8  BDF744           JSR   MULT1      ;MULTIPLY BY 2.44140625
F8EB  A701             STAA  $01,X
F8ED  E700             STAB  $00,X
F8EF  BDF76C           JSR   DECADJ     ;CONVERT TO A DECIMAL NUMBER
F8F2  CE4023           LDX   #CONVH
F8F5  5F               CLRB             ;CLEAR ↑B
F8F6  A601             LDAA  $01,X      ;ADD CONVL TO ITSELF,
F8F8  AB01             ADDA  $01,X      ;CONVERT TO A ↑B↑C↑D NUMBER
F8FA  19               DAA
F8FB  B74032           STAA  CONVL1     ;AND STORE IN CONVL1
F8FE  A600             LDAA  $00,X
F900  A900             ADCA  $00,X      ;ADD CONVH TO ITSELF PLUS CARRY
F902  19               DAA              ;CONVERT TO A ↑B↑C↑D NUMBER
F903  B74031           STAA  CONVH1     ;STORE IN CONVH1
F906  01               NOP
F907  C900             ADCB  #$00        ;ADD CARRY TO ↑B
F909  B64032           LDAA  CONVL1     ;ADD CONVL1 TO CONVL
F90C  AB01             ADDA  $01,X
F90E  19               DAA              ;CONVERT TO ↑B↑C↑D
F90F  A701             STAA  $01,X      ;STORE IN CONVL
F911  B64031           LDAA  CONVH1
F914  A900             ADCA  $00,X      ;ADD CONVH1 TO CONVH PLUS CARRY
F916  19               DAA              ;CONVERT TO ↑B↑C↑D NUMBER
F917  A700             STAA  $00,X      ;STORE IN CONVH
F919  C900             ADCB  #$00        ;ADD CARRY TO ↑B
F91B  17               TBA              ;PUT ↑B IN ↑A
F91C  8B08             ADDA  #$08        ;ADD 8 TO ↑A
F91E  19               DAA              ;CONVERT TO A ↑B↑C↑D DIGIT
F91F  36               PSHA
F920  8410             ANDA  #$10        ;CHECK FOR A LEADING 1
F922  2707             BEQ   ZERO       ;IF NOT, GO TO ZERO
F924  32               PULA
F925  BDF57A           JSR   BYTOUT     ;IF SO, PRINT 2 LEADING DIGITS
F928  7EF876           JMP   PRTALL
F92B  BDF600  ZERO     JSR   PSP        ;PRINT A SPACE
F92E  32               PULA
F92F  8A30             ORAA  #$30        ;CONVERT ↑A TO ASCII
F931  BDF2A0           JSR   OUTA       ;PRINT THE DIGIT
F934  7EF876           JMP   PRTALL
F937  A601    NEGNUM   LDAA  $01,X
F939  E600             LDAB  $00,X
F93B  40               NEGA             ;GET THE MAGNITUDE OF THE
F93C  50               NEGB             ;NEGATIVE NUMBER
F93D  5A               DECB
F93E  A701             STAA  $01,X
F940  E700             STAB  $00,X
F942  862D             LDAA  #$2D
F944  BDF2A0           JSR   OUTA       ;PRINT THE MINUS SIGN
F947  7EF856           JMP   TOGETR     ;JUMP TO TOGETR
              ******************
              **THIS SUBROUTINE SHIFTS CONSECUTIVE MEMORY
```

```
                    *LOCATIONS (SPECIFIED BY THE ↑I↑R) FOUR TIMES
                    *TO THE RIGHT.
F94A    BDF575  RSHIFT4 JSR     SHIFTR
F94D    BDF575          JSR     SHIFTR
F950    BDF575          JSR     SHIFTR
F953    BDF575          JSR     SHIFTR
F956    39              RTS
                    *********************
                    **↑THIS SUBROUTINE CONVERTS A 12 BIT DATA
                    *WORD IN [↑A↑B] TO A 8 BIT DATA WORD IN ↑A,
                    *WITH ROUNDING.
F957    FF4028  MOVEIT  STX     TEMPIR          ;SAVE ↑I↑R
F95A    CE0004          LDX     ¬$$04
F95D    54      MOVLOP  LSRB                    ;SHIFT ↑B
F95E    46              RORA                    ;SHIFT ↑A
F95F    09              DEX                     ;DECREMENT ↑I↑R
F960    26FB            BNE     MOVLOP          ;TEST IF DONE
F962    8900            ADCA    ¬$$00            ;ROUND THE NUMBER IF DONE
F964    C900            ADCB    ¬$$00
F966    FE4028          LDX     TEMPIR          ;RESORE ↑I↑R
F969    39              RTS
                    *********************
                    **↑THIS SUBROUTINE SHIFTS [↑B↑A] TO THE
                    *RIGHT 4 TIMES.
F96A    FF4028  LEFTSH  STX     TEMPIR
F96D    CE0004          LDX     ¬$$04
F970    48      LSH     ASLA
F971    59              ROLB
F972    09              DEX
F973    26FB            BNE     LSH
F975    FE4028          LDX     TEMPIR
F978    39              RTS
                    *********************
                    **↑THIS ROUTINE IS EXECUTED WHENEVER
                    *A HARDWARE INTERRUPT OCCURS(↑N↑O↑T A
                    *NON-MASKABLE INTERRUPT).  ↑ANY TIME
                    *THE ↑A↑C↑I↑A RECEIVES A CHARACTER THIS
                    *ROUTINE IS EXECUTED.
F979    B6D800  IRQ     LDAA    CNTRL           ;LOAD ↑A WITH STATUS
F97C    8470            ANDA    ¬SEMASK          ;TEST FOR A TRANSMISSION ERROR
F97E    270A            BEQ     NOERR           ;IF NO ERROR BRANCH
F980    863F            LDAA    ¬$$3F
F982    BDF2A0          JSR     OUTA            ;IF AN ERROR OCCURED PRINT
F985    B6D801          LDAA    DATA            ;A QUESTION MARK
F988    2028            BRA     IRQHERE         ;FINISH THE ROUTINE
F98A    B6D801  NOERR   LDAA    DATA
F98D    BDF2A0          JSR     OUTA            ;PRINT THE RECEIVED CHARACTER
F990    9712            STAA    RECEIV          ;STORE THE RECEIVED CHARACTER
F992    D613    ENDING  LDAB    SIRQ
F994    CA04            ORAB    ¬$$04
F996    8118            CMPA    ¬$$18            ;IS THE CHARACTER AN ↑E↑S↑C↑A↑P↑E
F998    2719            BEQ     ESC             ;IF YES BRANCH TO ESC
F99A    8113            CMPA    ¬$$13            ;IS THE CHARACTER A CONTROL ↑S
F99C    2612            BNE     STATSO          ;IF NOT BRANCH TO STATSO
F99E    BDF60A  FAKE    JSR     FAKERI          ;PULL ANYTHING AN ↑R↑T↑S WOULD
F9A1    CE0000          LDX     ¬$0000
F9A4    DF00            STX     BACKUP          ;CLEAR BACKUP
```

```
F9A6    7F000A            CLR     WHERE              ;CLEAR WHERE
F9A9    BDF555            JSR     LFCR               ;DO A ↑L↑F ↑C↑R
F9AC    0E                CLI                        ;ENABLE FURTHER ↑I↑R↑Q↑7S
F9AD    7EFCCB            JMP     DISDON             ;RETURN TO USER PAGE
F9B0    D713     STATSO   STAB    SIRQ               ;INDICATE AN ↑I↑R↑Q OCCURED
F9B2    3B       IRQHERE  RTI                        ;RETURN TO PREVIOUS WORK
F9B3    7D0000   ESC      TST     BACKUP             ;TEST BACKUP
F9B6    27F8              BEQ     STATSO             ;IF ZERO THEN DONE
F9B8    BDF60A            JSR     FAKERI             ;IF NOT ZERO PULL FROM THE
                                                     ;STACK LIKE A ↑R↑T↑I
F9BB    32       ESCLOP   PULA                       ;THIS LOOP FIXES THE STACK
F9BC    B74023            STAA    CONVH
F9BF    33                PULB
F9C0    F74024            STAB    CONVL
F9C3    FE4023            LDX     CONVH
F9C6    8CFC79            CPX     ¬$STRIP             ;CHECK FOR THE ADDRESS OF STRIP
F9C9    26F0              BNE     ESCLOP             ;IF NOT PULL MORE OFF THE STACK
F9CB    37                PSHB                       ;IF STRIP REPLACE ON STACK
F9CC    36                PSHA
F9CD    BDF5F3            JSR     PSPACE             ;SKIP A SPACE
F9D0    CEF7BA            LDX     ¬$DEL
F9D3    BDF717            JSR     PRINT              ;PRINT A ↑7*↑D↑E↑L*↑7
F9D6    BDF555            JSR     LFCR               ;SKIP A LINE
F9D9    DE00              LDX     BACKUP
F9DB    0E                CLI                        ;ENABLE INTERRUPT
F9DC    6E00              JMP     $00,X              ;JUMP BACK A LINE
                 **********************
                 *↑THIS SUBROUTINE GENERATES A
                 *0.50 SECOND DELAY.
F9DE    CEA480   DELAY    LDX     ¬$A480
F9E1    09       DLOOP    DEX
F9E2    26FD              BNE     DLOOP
F9E4    39                RTS
                 *********************
                 *↑THIS SUBROUTINE GENERATES A
                 *ONE MILLISECOND DELAY.
F9E5    DF02     MDELAY   STX     TEMPI3
F9E7    CE0054            LDX     ¬$54
F9EA    09       MDLOOP   DEX
F9EB    26FD              BNE     MDLOOP
F9ED    DE02              LDX     TEMPI3
F9EF    39                RTS
                 ************************
                 ****************
                 *↑THIS ROUTINE RESPONDS TO ALL INTERRUPTS
                 *FROM THE CLOCK BOARD.  ↑USUALLY THIS MEANS
                 *A READING IS TO BE TAKEN OR WARMUP POWER
                 *IS TO BE TURNED ON.
F9F0    86F7     NMI      LDAA    ¬$F7                ;OPTIONAL
F9F2    B7E002            STAA    CLOCKB             ;OPTIONAL
F9F5    8604              LDAA    ¬$04                ;OPTIONAL
F9F7    B7E003            STAA    CLOCKB+1           ;OPTIONAL
F9FA    BDF61E            JSR     OFFCLK             ;PREVENT MULTIPLE INTERRUPTS
F9FD    BDF9DE            JSR     DELAY              ;LET  SENSORS STABLIZE
FA00    BDF629            JSR     ENMEM              ;ENABLE MEMORY IN CASE IT IS OFF
FA03    B64001            LDAA    MONTH              ;THE STORE AND LOADS SAVE
FA06    9704              STAA    TMONTH             ;THE VARIABLES USED FOR
```

```
FA00   B64002             LCAA    DAY          ;SEQUENTIAL READINGS
FA03   9705               STAA    TDAY
FA05   B64003             LCAA    TIMEH
FA10   9706               STAA    TTIMEH
FA12   B64004             LCAA    TIMEL
FA15   9707               STAA    TTIMEL
FA17   B64033             LCAA    NUMLOW
FA1A   9708               STAA    TNUMLOW
FA1C   B64034             LCAA    NUMHI
FA1F   9709               STAA    TNUMHI
FA21   BDF5B3             JSR     RTIME        ;RESTORE ALL VARIABLES TO
                                               ;THEIR TRUE VALUES
FA24   BDF2E1   UP        JSR     UPTIME       ;UPDATE THE TIME
FA27   B64000             LCAA    STATUS
FA2A   8408               ANDA    #$08          ;CHECK IF MEASUREMENTS ARE ALL TAKEN
FA2C   2707               EEQ     CHKW         ;IF NOT GO TO CHKW
FA2E   C6FF               LCAB    #$FF          ;RESET CLOCK TO LARGEST INPUT
FA30   86FF               LCAA    #$FF
FA32   7EFB17             JMP     SLEEPY       ;SLEEPY RESETS CLOCK TO[^A^B]
FA35   B64000   CHKW      LCAA    STATUS
FA38   8401               ANDA    #$01          ;DO READINGS START TODAY^/
FA3A   271E               EEQ     WARM         ;IF YES BRANCH TO WARM
FA3C   BDF3CF             JSR     PSCMPW       ;COMPARE STARTING DATE TO TODAY
                                               ;24 HOURS OR <24 HOURS AWAY
FA3F   260A               BNE     NMITODY
FA41   8605               LCAA    #$05          ;LOAD 24 HOURS
FA43   C6A0               LCAB    #$A0
FA45   BDF63F             JSR     CLKSET
FA48   7EFB18             JMP     SHUTOFF
FA4B   B64013   NMITODY   LCAA    STIMEH       ;LOAD STARTING TIME
FA4E   F64014             LCAB    STIMEL
FA51   BDF63F             JSR     CLKSET
FA54   7A4000             CEC     STATUS
FA57   7EFB18             JMP     SHUTOFF
FA5A   B64000   WARM      LCAA    STATUS
FA5D   8402               ANDA    #$02          ;CHECK IF LAST POWER OFF WAS
                                               ;FOR WARM-UP OR IF WARM-UP=0
FA5F   261E               BNE     MEASUR       ;IF IT WAS FOR WARM-UP GOTO MEASUR
                                               ;AND TAKE SOME READINGS
FA61   B64000             LCAA    STATUS
FA64   8B02               ACDA    #$02          ;INDICATE POWER OFF FOR WARM-UP
FA66   B74000             STAA    STATUS
FA69   4F                 CLRA                 ;CLEAR ^A
FA6A   F64016             LCAB    WTIME        ;GET THE WARM-UP TIME
FA6D   BDF63F             JSR     CLKSET       ;SET THE CLOCK TO WARM-UP TIME
FA70   BDF636             JSR     CLICLK       ;ENABLE THE CLOCK TO INTERRUPT
FA73   BDF4A3             JSR     DISMEM       ;DISABLE THE MEMORY
FA76   BDF368             JSR     WRMPOW       ;TURN OFF ALL POWER EXCEPT WARM-UP
FA79   BDF9DE             JSR     DELAY        ;WAIT FOR RELAYS TO WORK
FA7C   7EFB27             JMP     RESUME
FA7F   7F402C   MEASUR    CLR     SENNUM       ;SENNUM POINTS TO A PORT
FA82   FE4025             LCX     MDATAH       ;^I^R HOLDS WHERE TO PUT DATA
FA85   BDFB71   TEMPDO    JSR     READ         ;READ A PORT INTO [^B^A]
FA88   BDF957             JSR     MOVEIT       ;ROUND THE READING TO 8 BITS
FA8B   A700               STAA    $00,X        ;STORE THE RESULT IN MEMORY
FA8D   08                 INX                  ;INCREMENT ^I^R
FA8E   7C402C             INC     SENNUM       ;LOOK AT NEXT PORT
```

```
FA91   B6402C          LDAA   SENNUM
FA94   8102            CMPA   ¬$$02              ;ARE PORT 1 AND 2 DONE↑/
FA96   26ED            BNE    TEMPDO             ;IF ↑N↑O↑T BRANCH BACK
FA98   BDFB71          JSR    READ               ;TAKE NEXT(AIR P.) READING
FA9B   BDF96A          JSR    LEFTSH             ;ARRANGE 12 BITS FOR MEMORY
FA9E   E700            STAB   $00,X              ;STORE HIGH ORDER 8 BITS
FAA0   08              INX                       ;INCREMENT ↑I↑R
FAA1   A700            STAA   $00,X              ;STORE LOW ORDER 4 BITS
FAA3   B6402B          LDAA   NPORTS             ;NUMBER OF PORTS TO READ
FAA6   8002            SUBA   ¬$$02              ;SUBTRACT 2 ALREADY READ
FAA8   B7402D          STAA   FULLR              ;STORE THAT IN TEMP LOC.
FAAB   BDFC15          JSR    SETUP              ;INITIALIZE SETUP
FAAE   7A402D  MORFUL  DEC    FULLR              ;DEC FROM LAST PORT READ
FAB1   2723            BEQ    INCONE             ;BRANCH IF DONE
FAB3   BDFBE7          JSR    WHICHP             ;FIND NEXT PORT TO READ
FAB6   BDFB71          JSR    READ               ;READ THE PORT
FAB9   EA00            ORAB   $00,X              ;↑O↑R HIGH ORDER 4 BITS WITH
                                                 ;WHAT IS ALREADY THERE
FABB   E700            STAB   $00,X              ;STORE THE COMBINATION
FABD   08              INX                       ;INCREMENT ↑I↑R
FABE   A700            STAA   $00,X              ;STORE THE LOWER 8 BITS
FAC0   7A402D          DEC    FULLR              ;DECREMENT THE ¬$ OF READINGS
FAC3   2711            BEQ    INCONE             ;BRANCH IF DONE
FAC5   08              INX                       ;INCREMENT ↑I↑R
FAC6   BDFBE7          JSR    WHICHP             ;FIND THE NEXT PORT
FAC9   BDFB71          JSR    READ               ;READ IT
FACC   BDF96A          JSR    LEFTSH             ;SHIFT READING
FACF   E700            STAB   $00,X              ;STORE HIGH ORDER 8 BITS
FAD1   08              INX                       ;INCREMENT ↑I↑R
FAD2   A700            STAA   $00,X              ;STORE LOW ORDER 8 BITS
FAD4   20D8            BRA    MORFUL             ;BRANCH AND CHECK FOR MORE
FAD6   08      INCONE  INX                       ;INCREMENT ↑I↑R
FAD7   FF4025          STX    MDATAH             ;STORE BACK IN MDATAH, THIS IS THE
                                                 ;NEXT WORD FOR STORING DATA
FADA   B64018          LDAA   WTIME
FADD   2708            BEQ    OKSTAT             ;IF NO WARM-UP TIME STATUS WORD IS ↑O↑K
FADF   B64000          LDAA   STATUS
FAE2   84FD            ANDA   ¬$$FD              ;RESET WARM-UP BIT
FAE4   B74000          STAA   STATUS             ;STORE BACK INTO  MEMORY
FAE7   BDF4B1  OKSTAT  JSR    PFCOMP             ;COMPARE TODAY TO FINAL DAY
FAEA   2722            BEQ    AWAY               ;IF NOT EQUAL BRANCH TO AWAY
FAEC   F64019          LDAB   FTIMEH
FAEF   B6401A          LDAA   FTIMEL             ;[↑B↑A]=STOPPING  TIME
FAF2   B04008          SUBA   PTIMEL
FAF5   F24007          SBCB   PTIMEH             ;[↑B↑A]=STOP TIME-PRESENT TIME
FAF8   2C0E            BGE    AWAY1              ;IF NEGATIVE GOTO LETSGO
FAFA   01      LETSGO  NOP
FAFB   B64000          LDAA   STATUS
FAFE   8A08            ORAA   ¬$$08              ;INDICATE READINGS ALL TAKEN
FB01   B74000          STAA   STATUS
FB04   86FF            LDAA   ¬$$FF
FB06   16              TAB                       ;↑A↑B=FF
FB07   200F            BRA    SLEEPY             ;GO STUFF TIME IN CLOCK
FB09   5D      AWAY1   TSTB
FB0A   2603            BNE    AWAY               ;IF  ↑B<>0 THEN GOTO AWAY
FB0C   4D              TSTA
FB0D   27EC            BEQ    LETSGO             ;IF ↑A=0 THEN GOTO AWAY1
```

```
FB0E    F64016   AWAY     LDAB    LFREQ
FB11    B64015            LDAA    HFREQ          ;[↑A↑B]=FREQ OF MEASUREMENTS
FB14    BDF386            JSR     SUBWTI         ;SUBTRACT THE WARM-UP TIME
FB17    01       SLEEPY   NOP
FB18    BDF63F            JSR     CLKSET         ;STUFF THE CLOCK WITH TIME
FB1B    BDF636   SHUTOFF  JSR     CLICLK         ;ALLOW THE CLOCK TO INTERRUPT
FB1E    BDF4A3            JSR     DISMEM         ;DISABLE THE MEMORY
FB21    BDF358            JSR     POWOFF         ;TURN THE POWER OFF
FB24    BDF9DE            JSR     DELAY          ;WAIT FOR THE RELAYS
FB27    960A     RESUME   LDAA    WHERE
FB29    2612              BNE     GOBACK         ;IF NOT 0, SOMETHING WAS INTERRUPTED
FB2B    8680              LDAA    ¬$$80
FB2D    BDF629            JSR     ENMEM
FB30    B4400C            ANDA    STATUS
FB33    BDF4A3            JSR     DISMEM
FB36    4D                TSTA
FB37    2703              BEQ     GOODBYE        ;IF NOT EQUAL, A DUMP WAS IN PROGRESS
                                                 ;THEREFORE MEMORY MUST BE LEFT ON
FB39    BDF629            JSR     ENMEM
FB3C    3B       GOODBYE  RTI
FB3D    2E2B     GOBACK   BGT     FIXCUR         ;WAS CURRENT OR SEQUENTIAL
                                                 ;READINGS INTERRUPTED↑/
FB3F    BDF60A            JSR     FAKERI         ;IF SEQUENTIAL FAKE ↑R↑T↑I
FB42    BDF629            JSR     ENMEM          ;ENABLE MEMORY
FB45    9604              LDAA    TMONTH         ;RESTORE VARIABLES TO LAST LINE PRINTED
FB47    B74001            STAA    MONTH
FB4A    9605              LDAA    TDAY
FB4C    B74002            STAA    DAY
FB4F    9606              LDAA    TTIMEH
FB51    B74003            STAA    TIMEH
FB54    9607              LDAA    TTIMEL
FB56    B74004            STAA    TIMEL
FB59    9608              LDAA    TNUMLOW
FB5B    B74033            STAA    NUMLOW
FB5E    9609              LDAA    TNUMHI
FB60    B74034            STAA    NUMHI
FB63    BDF555            JSR     LFCR           ;SKIP TO NEXT LINE
FB66    0E                CLI                    ;CLEAR INTERRUPT
FB67    7EFE56            JMP     RESSEQ         ;JUMP BACK TO PRINT NEW LINE
FB6A    BDF60A   FIXCUR   JSR     FAKERI         ;DO A FAKE ↑R↑T↑I FOR CURRENT READINGS
FB6D    0E                CLI                    ;CLEAR INTERRUPT
FB6E    7EFDF1            JMP     LEZTE          ;JUMP BACK TO CURRENT READINGS
FB71    FF4035   READ     STX     TMPIR          ;SAVE ↑I↑R
FB74    BDF6DA            JSR     ADCVAL         ;THROW AWAY FIRST READING
FB77    BDF6DA            JSR     ADCVAL         ;TAKE A GOOD READING
FB7A    8608              LDAA    ¬$NUMAVR        ;TELLS HOW MANY READINGS TO AVERAGE
FB7C    B7401D            STAA    TEMP2          ;STORE NUMAVR
FB7F    B7401F            STAA    TEMP4          ;STORE NUMAVR
FB82    4F                CLRA
FB83    B7401C            STAA    TEMP1          ;STORE 0
FB86    B7401E            STAA    TEMP3          ;STORE 0
FB89    B64024            LDAA    CONVL
FB8C    F64023            LDAB    CONVH          ;[↑B↑A]=READING
FB8F    FE401C            LDX     TEMP1          ;↑I↑R=NUMAVR
FB92    09       SUMCON   DEX                    ;DECREMENT ↑I↑R
FB93    270E              BEQ     SUMDON         ;IF ZERO THEN SUM IS DONE
FB95    BDF9E5            JSR     MDELAY         ;DELAY BEFORE NEXT READING
```

```
FB98   BDF6DA          JSR    ADCVAL         ;TAKE NEXT READING
FB9B   BB4024          ADDA   CONVL
FB9E   F94023          ADCB   CONVH          ;[↑B↑A]=[↑B↑A]+READING
FBA1   20EF            BRA    SUMCON         ;BRANCH BACK FOR MORE READINGS
FBA3   CE0004  SUMDON  LDX    #$$04           ;=1+LN(NUMAVR)/LN(2)
FBA6   09      AVRDIV  DEX                   ;DECREMENT ↑I↑R
FBA7   2704            BEQ    AVGDON         ;IF EQUAL ZERO THAN DONE
FBA9   54              LSRB
FBAA   46              RORA                  ;DIVIDE SUM BY 2
FBAB   20F9            BRA    AVRDIV         ;BRANCH BACK FOR MORE DIVIDES
FBAD   FE4035  AVGDON  LDX    TMPIR          ;AVERAGE IS CALCULATED
FBB0   39              RTS                   ;RESTORE ↑I↑R AND RETURN
               ************************
               *↑SUBROUTINE TIMEIT CALCULATES THE NEXT TIME
               *THE CLOCK HAS TO TURN ON.  ↑IF THE SYSTEM
               *DOESN↑7T HAVE TO TAKE READINGS TILL TOMORROW
               *THEN MIDNIGHT MINUS THE PRESENT TIME MINUS
               *THE WARMUP TIME IS STUFFED INTO THE CLOCK.
               *↑IF THE READINGS START TODAY THEN THE STARTING
               *TIME MINUS THE PRESENT TIME MINUS THE WARM-
               *UP TIME IS STUFFED INTO THE CLOCK.  ↑THE BIT
               *IN THE STATUS WORD THAT INDICATES IF THE
               *SYSTEM IS TO START TAKING READINGS NEXT TIME
               *IT ↑,COMES UP↑, IS APPROPRIATELY SET OR RESET.
FBB1   2610    TIMEIT  BNE    TODAY          ;BRANCH IF STARTS TODAY
FBB3   BDF206          JSR    MIDNIG         ;[↑A↑B]=MIDNIGHT-PRESENT TIME
FBB6   BDF386          JSR    SUBWTI         ;SUBTRACT WARMUP TIME
FBB9   BDF63F          JSR    CLKSET         ;STUFF INTO THE CLOCK
FBBC   BDF629          JSR    ENMEM          ;ENABLE MEMORY
FBBF   4F              CLRA
FBC0   4C              INCA
FBC1   201D            BRA    OFFSET         ;BRANCH TO SET STATUS
FBC3   B64013  TODAY   LDAA   STIMEH         ;IT STARTS TODAY
FBC6   F64014          LDAB   STIMEL
FBC9   F04006          SUBB   PTIMEL
FBCC   B24007          SBCA   PTIMEH         ;[↑A↑B]=START TIME -PRESENT TIME
FBCF   BDF386          JSR    SUBWTI         ;SUBTRACT WARMUP TIME
FBD2   BDF63F          JSR    CLKSET         ;STUFF INTO THE CLOCK
FBD5   BDF629          JSR    ENMEM          ;ENABLE MEMORY
FBD8   4F              CLRA
FBD9   F64016          LDAB   WTIME
FBDC   2602            BNE    OFFSET
FBDE   8B02            ADDA   #$$02           ;SET STATUS FOR NO WARMUP TIME
FBE0   B74000  OFFSET  STAA   STATUS         ;STORE STATUS WORD
FBE3   BDF4A3          JSR    DISMEM         ;TURN OFF MEMORY
FBE6   39              RTS                   ;RETURN TO CALLER
               **********************
               *↑THIS ROUTINE RETURNS THE NEXT ADDITIONAL
               *PORT SELECTED IN ↑B AND SENNUM.  ↑IF NO
               *MORE PORTS ARE SELECTED, IT RETURNS FF.
               *↑SUBROUTINE SETUP MUST BE CALLED BEFORE
               *WHICHP IF FIRST USED.
FBE7   F6402F  WHICHP  LDAB   WHICH1
FBEA   B64030          LDAA   WHICH2
FBED   36      DONTNO  PSHA
FBEE   8401            ANDA   #$$01           ;CHECK IF WHICH1 IS NUMBER OF SELECTED
FBF0   260E            BNE    IKNOW          ;PORT IF SO, GOTO I KNOW
```

```
FBF2    32      GIVEUP  PULA
FBF3    5C              INCB                            ;IF NOT, INCREMENT ↑B
FBF4    46              RORA                            ;SHIFT ↑A RIGHT
FBF5    B74030          STAA    WHICH2                  ;STORE IN WHICH2
FBF8    C107            CMPB    ¬$$07                    ;CHECK IF THIS IS AFTER LAST PORT
FBFA    2FF1            BLE     DONTNO                  ;IF NOT, GO TO DONTNO
FBFC    C6FF            LDAB    ¬$$FF                    ;IF SO, LOAD ↑B WITH FF
FBFE    2011            BRA     DEPART
FC00    32      INNCW   PULA
FC01    764030          ROR     WHICH2                  ;SHIFT WHICH2 RIGHT
FC04    5C              INCB                            ;INCREMENT ↑A
FC05    C108            CMPB    ¬$$08                    ;CHECK IF THIS IS AFTER THE LAST PORT
FC07    2F04            BLE     BARK                    ;IF NOT, GO TO BARK
FC09    C6FF            LDAB    ¬$$FF                    ;IF SO, LOAD ↑B WITH FF
FC0B    2004            BRA     DEPART
FC0E    F7402F  BARK    STAB    WHICH1
FC10    5A              DECB
FC11    F7402C  DEPART  STAB    SENNUM
FC14    39              RTS
                *********************
                *↑THIS ROUTINE INITIALIZES WHICH1
                *AND WHICH2 FOR USE IN WHICHP
FC15    B6402A  SETUP   LDAA    PORTBT
FC18    B74030          STAA    WHICH2
FC1B    7F402F          CLR     WHICH1
FC1E    39              RTS
                *********************
                *↑THIS ROUTINE INITIALIZES THE DATA LOGGER
                *IF YOU PRESS ↑7D↑7 AND RESPOND WITH A YES.
FC1F    BDF555  INIT    JSR     LFCR                    ;SKIP A LINE
FC22    CEF0C2          LDX     ¬$PINIT                  ;WRITES ↑7DO YOU WANT TO ...
FC25    BDF717          JSR     PRINT                   ;PRINTS THE MESSAGE
FC28    CEF0D2          LDX     ¬$YESNO
FC2B    BDF717          JSR     PRINT                   ;PRINTS ↑7(YES,NO)↑7
FC2E    BDF131          JSR     CHYES                   ;TESTS FOR A YES OR A NO
FC31    2737            BEQ     FIRSTON                 ;BRANCH IF ANSWER IS YES
FC33    BDF555          JSR     LFCR                    ;SKIP A LINE
FC36    206A            BRA     DISP                    ;RETURN TO CHOICE PAGE
                ********************
                *↑THIS IS WERE THE PROCESSOR STARTS WHENEVER
                *IT IS FIRST TURNED ON.  ↑THE FIRST FEW INSTRUC-
                *TIONS BASICLY INITIALIZE ANY PARTS OF THE
                *SYSTEM THAT NEED TO BE REINITIALIZED.
                *(↑BECAUSE THE TIME IN THE CLOCK MIGHT BE
                *CHANGED WHEN THE SYSTEM IS TURNED ON
                *THE SYSTEM RELOADS THE TIME IN THE CLOCK.)
                *****↑N↑O↑T↑E↑;  ↑IF THE POWER IS OFF AND THE
                *PROCESSOR TURNS ON IT JUMPS TO THIS SECTION
                *OF CODE IMMATERIAL OF WHO TURNED ON THE
                *POWER (THE SYSTEM ITSELF OR THE USER)↑.
FC38    8E007F  RESTART LDS     ¬$STKPTR                 ;INITIALIZE THE STACK POINTER
FC3B    7F000A          CLR     WHERE
FC3E    BDF697          JSR     PIAS                    ;INITIALIZE THE  ↑P↑I↑A↑7↑S
FC41    BDF629          JSR     ENMEM                   ;ENABLE THE MEMORY
FC44    BDF61E          JSR     OFFCLK                  ;DISABLE THE CLOCK↑7S INTERRUPT
FC47    B64009          LDAA    LMINH
FC4A    2705            BEQ     HIGHZ                   ;CHECK IF LMINH = 0
```

```
FC4C   F6400A            LDAB    LMINL
FC4F   2005              BRA     RELOAD          ;TIME IN CLOCK WAS NOT 0.
FC51   F6400A   HIGHZ    LDAB    LMINL
FC54   2703              BEQ     REDONE          ;BRANCH IF TIME IN CLOCK WAS 0
FC56   BDF63F   RELOAD   JSR     CLKSET          ;RESET THE TIME IN THE CLOCK
FC59   BDF636   REDONE   JSR     CLICLK          ;ENABLE THE CLOCK↑7S INTERRUPT
FC5C   BDF4A3            JSR     DISMEM          ;DISABLE THE MEMORY
FC5F   0E                CLI                     ;ALLOW KEYBOARD INTERRUPTS
FC60   BDF555            JSR     LFCR
FC63   CE0000            LDX     ¬$0000
FC66   DF00              STX     BACKUP          ;INITIALIZE BACKUP TO ZERO
FC68   2038     DISPON   BRA     DISP            ;GO TO THE OPTIONS PAGE
                *********************
                *↑THE PROGRAM JUMPS HERE IF THE SYSTEM
                *IS TO BE INITIALIZED (JUMPS FROM INIT).
                *↑FIRSTON INITIALIZES DATA COLLECTING
                *PARAMTERS, SET THE STATUS WORD, AND
                *PUTS THE FIRST TIME IN THE CLOCK.
FC6A   BDF61E   FIRSTON  JSR     OFFCLK          ;TURN OFF THE CLOCKS INTERRUPT
FC6D   BDF629            JSR     ENMEM           ;ENABLE MEMORY
FC70   CE4040            LDX     ¬$STARTM
FC73   FF4025            STX     MDATAH          ;REPOSITION DATA POINTER TO
                                                 ;THE FIRST WORD
FC76   BDF160            JSR     INDAT           ;GET THE INITIALIZATION DATA
FC79   BDF261   STRIP    JSR     PSCOMP          ;↑IS TODAY STARTING DAY↑/
FC7C   BDFB61            JSR     TIMEIT          ;LET TIMEIT STUFF THE CLOCK
FC7F   BDF629            JSR     ENMEM           ;ENABLE MEMORY
FC82   B6402B            LDAA    NPORTS          ;LOAD THE ¬$ OF READINGS TO BE TAKEN
                                                 ;EACH TIME THE PROCESSOR COMES UP
FC85   C604              LDAB    ¬$$04
FC87   C003              SUBA    ¬$$03
FC89   270B              BEQ     GOTIT           ;IF EQUAL THEN WE ARE DONE
FC8B   5C       OMYGOD   INCB                    ;ODD ¬$ OF EXTRA PORTS
FC8C   4A                DECA
FC8D   2707              BEQ     GOTIT           ;IF EQUAL TO 0 WE ARE DONE
FC8F   5C                INCB
FC90   5C                INCB                    ;EVEN ¬$ OF EXTRA PORTS
FC91   4A                DECA
FC92   2702              BEQ     GOTIT           ;IF ↑A=0 WE ARE DONE
FC94   20F5              BRA     OMYGOD          ;CHECK FOR MORE
FC96   F7402E   GOTIT    STAB    INCIR           ;INCIR IS THE ¬$ OF WORDS EACH
                                                 ;SET OF MEASUREMENTS USES OF MEMORY
FC99   BDF636            JSR     CLICLK          ;ENABLE THE CLOCK INTERRUPT
FC9C   BDF4A3            JSR     DISMEM          ;DISABLE THE MEMORY
FC9F   BDF356   PO       JSR     POWOFF          ;TURN THE POWER OFF
                 ****************
FCA2   9613     DISP     LDAA    SIRQ
FCA4   8404              ANDA    ¬$$04
FCA6   2603              BNE     CDISP           ;BRANCH ONLY IF AN INTERRUPT FROM
                                                 ;THE KEYBOARD HAS OCCURRED.
FCA8   7EFC68            JMP     DISPON          ;OTHERWISE LOOP AGAIN
FCAB   01       CDISP    NOP
FCAC   BDF555            JSR     LFCR
FCAF   BDF555            JSR     LFCR
FCB2   9612              LDAA    RECEIV          ;LOAD ↑A WITH LAST CHAR RECEIVED
FCB4   8141              CMPA    ¬$↑7A↑7
FCB6   271F              BEQ     JINPARM         ;IF AN ↑7A↑7 GOTO JINPARM
```

```
FCB8   8142             CMPA    ¬$↑7B↑7
FCBA   271E             BEQ     JCURRS          ;IF A ↑7B↑7 GOTO JCURRS
FCBC   8143             CMPA    ¬$↑7C↑7
FCBE   271D             BEQ     JSEQUEN         ;IF A ↑7C↑7 GOTO JSEQUEN
FCC0   8144             CMPA    ¬$↑7D↑7
FCC2   271C             BEQ     JINIT           ;IF A ↑7D↑7 THEN GOTO JINIT
FCC4   8145             CMPA    ¬$↑7E↑7
FCC6   271B             BEQ     JDUMP           ;IF AN ↑7E↑7 GOTO JDUMP
FCC8   BDF3FC           JSR     DISPCH          ;IF NONE REPRINT CHOICES
FCCB   9613    DISDON   LDAA    SIRQ
FCCD   84FB             ANDA    ¬$FB             ;CLEAR THE INTERRUPT SIGNAL
FCCF   9713             STAA    SIRQ
FCD1   BDF4A3           JSR     DISMEM          ;DISABLE MEMORY
FCD4   7EFC66           JMP     DISPDN          ;GO BACK AND WAIT FOR ANOTHER ↑I↑R↑Q
               *********************
               *↑THIS SECTION PROVIDES BRANCH
               *EXTENSIONS TO THE RIGHT SECTION.
FCD7   7EFD2F  JINPARM  JMP     INPARM
FCDA   7EFD9D  JCURRS   JMP     CURRS
FCDD   7EFE25  JSEQUEN  JMP     SEQUEN
FCE0   7EFC1F  JINIT    JMP     INIT
FCE3   7EFF1C  JDUMP    JMP     DUMP
               **********************
               *↑THIS SUBROUTINE PRINTS THE STARTING
               *DATE AND TIME.
FCE6   F64011  SPRINT   LDAB    SMONTH
FCE9   B64012           LDAA    SDAY
FCEC   BDF5EA           JSR     PDATE           ;PRINT THE DATE
FCEF   BDF5F5           JSR     PSPACE          ;SPACE
FCF2   F64013           LDAB    STIMEH
FCF5   B64014           LDAA    STIMEL
FCF8   BDF726           JSR     PTIME           ;PRINTS THE STARTING TIME
FCFB   BDF555           JSR     LFCR
FCFE   39               RTS
               *********************
               *↑THIS SUBROUTINE PRINTS THE STOPPING
               *DATE AND TIME.
FCFF   F64017  FPRINT   LDAB    FMONTH
FD02   B64018           LDAA    FDAY
FD05   BDF5EA           JSR     PDATE           ;PRINT THE STOPPING DATE
FD08   BDF5F5           JSR     PSPACE
FD0B   F64019           LDAB    FTIMEH
FD0E   B6401A           LDAA    FTIMEL
FD11   BDF726           JSR     PTIME           ;PRINT THE STOPPING TIME
FD14   BDF555           JSR     LFCR
FD17   39               RTS
               ********************
               *↑THIS SUBROUTINE PRINTS THE WARMUP
               *TIME.
FD18   B6401B  WPRINT   LDAA    WTIME
FD1B   BDF3E6           JSR     BCD             ;CHANGE THE TIME TO BCD
FD1E   BDF57A           JSR     BYTOUT          ;PRINT THE TIME
FD21   39               RTS
               *******************
               *↑THIS SUBROUTINE PRINTS THE FREQ-
               *UENCY OF MEASUREMENTS.
FD22   F64015  FRPRINT LDAB     HFREQ
```

```
FD25    B64016          LDAA    LFREQ
FD28    BDF726          JSR     PTIME           ;PRINT THE TIME
FD2B    BDF555          JSR     LFCR
FD2E    39              RTS
                ***********************
                *↑THIS SUBROUTINE PRINTS THE INPUT
                *PARAMTERS TO THE DATA LOGGER.
FD2F    BDF629  INPARM  JSR     ENMEM           ;ENABLE MEMORY
FD32    CEF022          LDX     #$START
FD35    BDF717          JSR     PRINT           ;PRINT ↑7START↑7
FD38    BDF5F5          JSR     PSPACE
FD3B    BDFCE6          JSR     SPRINT          ;PRINT THE STARTING DATE AND TIME
FD3E    CEF029          LDX     #$STOP
FD41    BDF717          JSR     PRINT           ;PRINT ↑7STOPPING↑7
FD44    BDF600          JSR     PSP             ;PRINT A SPACE
FD47    BDF5F5          JSR     PSPACE          ;PRINT 5 SPACES
FD4A    BDFCFF          JSR     FPRINT          ;PRINT THE STOPPING DATE AND TIME
FD4D    CEF051          LDX     #$WARMUP
FD50    BDF717          JSR     PRINT           ;PRINT ↑7WARMUP↑7
FD53    CEF05F          LDX     #$TI
FD56    BDF717          JSR     PRINT           ;PRINT ↑7TIME↑7
FD59    BDFD16          JSR     WPRINT          ;PRINT THE WARMUP TIME
FD5C    CEF065          LDX     #$MIN
FD5F    BDF717          JSR     PRINT           ;PRINT ↑7MIN↑7
FD62    BDF555          JSR     LFCR
FD65    CEF02F          LDX     #$FREQ
FD68    BDF717          JSR     PRINT           ;PRINT ↑7FREQUENCY OF ....↑7
FD6B    BDF5F5          JSR     PSPACE
FD6E    BDFD22          JSR     FRPRINT         ;PRINT THE FREQ. OF MEAS.
FD71    BDF4A3          JSR     DISMEM
FD74    7EFCCB          JMP     DISDON          ;RETURN TO OPTIONS
                **********************
                *↑THIS SUBROUTINE PRINTS THE PROPER
                *HEADING (DEPENDING ON THE PORTS
                *SELECTED) WHEN CALLED BY SEQUENTIAL
                *READINGS OR CURRENT SENSOR READINGS.
FD77    CEFFB7  SUBBA   LDX     #$HEAD2
FD7A    BDF717          JSR     PRINT
FD7D    CEFFCE          LDX     #$PORT
FD80    BDFC15          JSR     SETUP
FD83    BDFBE7  MORYET  JSR     WHICHP
FD86    C1FF            CMPB    #$FF
FD88    270F            BEQ     ZOOM
FD8A    CEFFCE          LDX     #$PORT
FD8D    BDF717          JSR     PRINT
FD90    5C              INCB
FD91    17              TBA
FD92    8A30            ORAA    #$30
FD94    BDF2A0          JSR     OUTA
FD97    20EA            BRA     MORYET
FD99    BDF555  ZOOM    JSR     LFCR
FD9C    39              RTS
                ****************************
                **********************
                *↑THIS SUBROUTINE DISPLAYS THE CURRENT
                *READINGS AT EACH PORT THAT WAS CHOSEN
                *DURING THE INITIALIZATION.  ↑ANY CHAR.
```

```
                        *TYPED AT THE KEYBOARD WILL DISPLAY
                        *THE CURRENT READINGS.  ↑TO EXIT FROM
                        *THIS SUBROUTINE YOU MUST HIT AN
                        *CONTROL ↑7S↑7.
FD9D     CEF0EF  CURRS    LDX      ¬$RS
FDA0     BDF717           JSR      PRINT           ;PRINT ↑7TYPE R FOR READ.....↑7
FDA3     BDF629           JSR      ENMEM           ;ENABLE MEMORY
FDA6     BDFD77           JSR      SUBBA           ;TYPE THE COLUMN HEADINGS
FDA9     BDF4A3           JSR      DISMEM          ;DISABLE THE MEMORY
FDAC     7C000A           INC      WHERE
FDAF     3E      MORER    WAI                      ;WAIT FOR A KEY PRESS
FDB0     BDF629           JSR      ENMEM           ;ENABLE MEMORY
FDB3     BDFC15           JSR      SETUP           ;INITAILIZE WHATPORT ROUTINE
FDB6     7F402C           CLR      SENNUM          ;START AT PORT 0
FDB9     BDF6DA           JSR      ADCVAL          ;TAKE A READING
FDBC     CE4023           LDX      ¬$CONVH          ;PUT READING IN ↑I↑R
FDBF     BDF56A           JSR      LSHFT4          ;SHIFT LEFT FOUR BITS
FDC2     BDF800           JSR      SENSCV          ;CONVERT AND PRINT THE TEMP.
FDC5     7C402C           INC      SENNUM          ;READ PORT 1
FDC8     CE4023           LDX      ¬$CONVH          ;PUT READING IN ↑I↑R
FDCB     BDF6DA           JSR      ADCVAL          ;TAKE THE READING
FDCE     BDF56A           JSR      LSHFT4          ;SHIFT LEFT FOUR BITS
FDD1     BDF800           JSR      SENSCV          ;CONVERT AND PRINT THE DEW POINT
FDD4     7C402C           INC      SENNUM          ;TAKE THE READING AT PORT 2
FDD7     BDF6DA           JSR      ADCVAL          ;TAKE THE AIR PRESSURE
FDDA     BDF800           JSR      SENSCV          ;CONVERT AND PRINT THE AIR PRESSURE
FDDD     BDFBE7  ERSTE    JSR      WHICHP          ;GET NUMBER OF ADDITIONAL PORT
FDE0     C1FF             CMPB     ¬$FF             ;IF IT IS FF THEY WE ARE DONE
FDE2     270D             BEQ      LEZTE           ;GOTO LEZTE IF DONE
FDE4     BDF6DA           JSR      ADCVAL          ;TAKE THE READING
FDE7     8604             LDAA     ¬$04             ;TELL SENSCV IT↑7S A VOLTAGE
FDE9     B7402C           STAA     SENNUM          ;STORE SENNUM
FDEC     BDF800           JSR      SENSCV          ;CONVERT AND PRINT THE VOLTAGE
FDEF     20EC             BRA      ERSTE           ;CHECK FOR MORE
FDF1     BDF555  LEZTE    JSR      LFCR
FDF4     BDF4A3           JSR      DISMEM          ;DISABLE THE MEMORY
FDF7     20B6             BRA      MORER           ;GO BACK AND WAIT FOR MORE
                        **********************
                        **↑THIS SUBROUTINE TAKES THE MEMORY LOCATION
                        *THAT IS IN THE ↑I↑R AND GIVES THE DATE AND
                        *TIME OF THE READING THAT IS STORED IN THAT
                        *MEMORY LOCATION.  ↑IF NO READING WAS TAKEN THEN
                        *THEN 000C IS STORED IN THE ↑I↑R.  ↑IF THE
                        *READING WAS TAKEN THEN THE DATE IS STORED
                        *IN MONTH AND DAY, AND THE TIME IS STORED IN
                        *TIMEH AND TIMEL.
FDF9     DF14    WHREAD   STX      SAVINGS         ;SAVE THE ↑I↑R
FDFB     BDF592           JSR      STIME           ;SAVE PMONTH THRU LMINH
FDFE     BDF4DC           JSR      STOPS           ;MOVE SMONTH-LFREQ INTO PMONTH-LMINL
FE01     CE4040           LDX      ¬$STARTM         ;START CHECKING AT THE START OF MEMORY
FE04     9C14    DAYLO    CPX      SAVINGS         ;IS THIS THE LOCATION IN QUESTION↑/
FE06     2710             BEQ      DDAY            ;IF IT IS BRANCH TO DDAY
FE08     BC4025           CPX      MDATAH          ;IF IT ISN↑7T THE CORRECT DAY
                                                   ;IS IT IN THE FUTURE↑/
FE0B     2708             BEQ      OVDAY           ;IF ITS↑7S A FUTURE READING
                                                   ;BRANCH TO OVDAY
FE0D     BDF4EA           JSR      IRLOOP          ;IF IT↑7S NONE OF THESE, INCREMENT
```

```
                                                    ;THE ↑I↑R APPROPRIATELY
FE10    BDF2E1            JSR    UPTIME             ;THEN CHANGE THE DATE AND TIME
FE13    20CF              BRA    DAYLO              ;BRANCH AND COMPARE AGAIN
FE15    CE0000  OVDAY     LDX    ¬$$0000             ;READING NEVER TAKEN
FE18    FF4021  DDAY      STX    TEMP12             ;SAVE ↑I↑R
FE1B    BDF378            JSR    UNSTOP             ;MOVE THE CORRECT DATE AND TIME
                                                    ;INTO MONTH,DAY,TIMEH,TIMEL
FE1E    BDF5B3            JSR    RTIME              ;RESTORE PMONTH-LMINL
FE21    FE4021            LDX    TEMP12             ;RESTORE THE ↑I↑R
FE24    39                RTS                       ;RETURN TO CALLER
                ************************
                *↑THIS SUBROUTINE TAKES A DATE AND TIME
                *AS INPUT AND ASKS HOW MANY READINGS
                *YOU WOULD LIKE PRINTED OUT.  ↑THE
                *SUBROUTINE THEN PRINTS OUT THE FIRST
                *READING ASKED FOR PLUS EVERY READING
                *AFTER THAT UNTIL THE NUMBER OF READ-
                *INGS THE USER SPECIFIED HAS BEEN
                *PRINTED.  ↑THE ROUTINE PRINTS NOTHING
                *IF A INVALID DATE OR TIME IS SPECIF-
                *IED.
FE25    BDF629  SEQUEN    JSR    ENMEM              ;ENABLE MEMORY
FE28    CEF022            LDX    ¬$START
FE2B    BDF717            JSR    PRINT              ;PRINT ↑7START↑7
FE2E    BDF246            JSR    GETDAT             ;GET THE DATE AND TIME
FE31    BDF555            JSR    LFCR
FE34    CEF78E            LDX    ¬$HOW
FE37    BDF717            JSR    PRINT              ;PRINT ↑7HOW MANY READINGS↑7
FE3A    CEF796            LDX    ¬$THREE
FE3D    BDF717            JSR    PRINT              ;PRINT ↑7(↑X↑X↑X)↑/↑7
FE40    BDF2AD            JSR    THRDIG             ;GET THE THREE DIGITS
FE43    B74033            STAA   NUMLOW             ;STORE THEM IN NUMLOW
FE46    F74034            STAB   NUMHI              ;AND NUMHI
FE49    BDF555            JSR    LFCR
FE4C    BDF600            JSR    PSP
FE4F    CEFFAC            LDX    ¬$HEAD1
FE52    BDF717            JSR    PRINT              ;PRINT ↑7DATE  TIME↑7
FE55    BDFD77            JSR    SUBBA              ;PRINT ↑7TEMP AIR PRE.....↑7
FE58    7A000A            DEC    WHERE
FE5B    BDF4F2  RESSEQ    JSR    RTAKEN             ;CHECK IF READING WAS TAKEN
FE5E    2603              BNE    SEQL               ;BRANCH IF READING WAS TAKEN
FE60    7EFF04            JMP    SEQDON             ;QUIT IF ↑I↑R=0000
FE63    BDFDF9  SEQL      JSR    WHREAD             ;FIND DATE AND TIME
FE66    BDF4F2            JSR    RTAKEN             ;FIND THE MEMORY LOCATION IT
                                                    ;IS STORED IN
FE69    2603              BNE    JUMP1              ;BRANCH IF FOUND
FE6B    7EFF04            JMP    SEQDON             ;QUIT
FE6E    F64001  JUMP1     LDAB   MONTH              ;↑B=MONTH
FE71    B64002            LDAA   DAY                ;↑A=MONTH
FE74    FF4026            STX    TEMPIR             ;SAVE ↑I↑R
FE77    BDF5EA            JSR    PDATE              ;PRINT THE DATE
FE7A    BDF600            JSR    PSP                ;SKIP A SPACE
FE7D    B64004            LDAA   TIMEL              ;↑A=TIMEL
FE80    F64003            LDAB   TIMEH              ;↑B=TIMEH
FE83    BDF726            JSR    PTIME              ;PRINT THE TIME
FE86    FE4026            LDX    TEMPIR             ;RESTORE ↑I↑R
FE89    F64026            LDAB   NPORTS             ;↑B=NUMBER OF READINGS
```

```
FE8C    C002                SUBB    ¬$$02           ;SUBTRACT TWO
FE8E    BDF600              JSR     PSP             ;SKIP A SPACE
FE91    A600                LDAA    $00,X           ;STORE READING IN ↑A
FE93    B74023              STAA    CONVH           ;STORE READING IN CONVH
FE96    7F402C              CLR     SENNUM          ;SET PORT TO 0
FE99    BDF800              JSR     SENSCV          ;CONVERT AND PRINT THE TEMP
FE9C    08                  INX                     ;INCREMENT THE ↑I↑R
FE9D    A600                LDAA    $00,X
FE9F    B74023              STAA    CONVH           ;STORE READING IN CONVH
FEA2    7C402C              INC     SENNUM          ;SET PORT TO 1
FEA5    BDF800              JSR     SENSCV          ;CONVERT AND PRINT THE DEW POINT
FEA8    08                  INX                     ;INCREMENT THE ↑I↑R
FEA9    5D          HITHER  TSTB                    ;ALL READINGS PRINTED FOR TIME
FEAA    2739                BEQ     YON1            ;IF YES BRANCH TO YON1
FEAC    5A                  DECB                    ;DECREMENT ¬$ OF READINGS TAKEN
FEAD    7C402C              INC     SENNUM          ;LOOKING AT NEXT PORT
FEB0    A600                LDAA    $00,X
FEB2    B74023              STAA    CONVH           ;STORE HIGH ORDER HALF IN CONVH
FEB5    A601                LDAA    $01,X
FEB7    B74024              STAA    CONVL           ;STORE LOW ORDER HALF IN CONVL
FEBA    FF4026              STX     TEMPIR          ;STORE THE ↑I↑R
FEBD    CE4023              LDX     ¬$CONVH          ;LET THE ↑I↑R POINT TO CONVH
FEC0    BDF94A              JSR     RSHIFT4         ;SHIFT CONVH+L RIGHT 4 BITS
FEC3    FE4026              LDX     TEMPIR          ;RESTORE ↑I↑R
FEC6    BDF800              JSR     SENSCV          ;CONVERT AND PRINT THE READING
FEC9    08                  INX                     ;INCREMENT THE ↑I↑R
FECA    5D                  TSTB                    ;HAS EACH READING BEEN PRINTED
FECB    2717                BEQ     YON2            ;IF YES BRANCH TO YON2
FECD    5A                  DECB                    ;DECREMENT ¬$ OF READINGS PRINTED
FECE    A601                LDAA    $01,X
FED0    B74024              STAA    CONVL           ;STORE LOW ORDER 8 BITS IN CONVL
FED3    A600                LDAA    $00,X
FED5    840F                ANDA    ¬$$0F            ;MASK OFF TOP 4 BITS
FED7    B74023              STAA    CONVH           ;STORE HIGH 8 BITS IN CONVH
FEDA    7C402C              INC     SENNUM          ;POINT TO NEXT PORT
FEDD    BDF800              JSR     SENSCV          ;CONVERT AND PRINT THE READING
FEE0    08                  INX                     ;INCREMENT THE ↑I↑R
FEE1    08                  INX                     ;DO IT TWICE
FEE2    20C5                BRA     HITHER          ;CHECK FOR MORE READINGS
FEE4    08          YON2    INX                     ;INCREMENT THE ↑I↑R
FEE5    BDF555      YON1    JSR     LFCP
FEE8    F64034              LDAB    NUMHI
FEEB    B64033              LDAA    NUMLOW
FEEE    8001                SUBA    ¬$$01            ;DECREMENT ¬$ OF READINGS PRINTED
FEF0    C200                SBCB    ¬$$00            ;WITH CARRY
FEF2    F74034              STAB    NUMHI           ;STORE IT BACK
FEF5    B74033              STAA    NUMLOW
FEF8    4D                  TSTA                    ;IS LOW ORDER 8 BITS ZERO
FEF9    2703                BEQ     HOP12           ;IF YES GO TEST ↑B
FEFB    7EFE63              JMP     SEQL            ;JUMP BACK FOR MORE READINGS
FEFE    5D          HOP12   TSTB                    ;TEST HIGH ORDER 8 BITS
FEFF    2703                BEQ     SEQDON          ;IF ZERO THEN QUIT
FF01    7EFE63              JMP     SEQL            ;IF NOT BRANCH BACK FOR MORE
FF04    BDF443      SEQDON  JSR     DISMEM          ;DISABLE MEMORY
FF07    7C000A              INC     WHERE
FF0A    7EFCCB              JMP     DISDON          ;RETURN TO OPTIONS PAGE
                    ***************************
```

```
FF0D   DF02     DELAY1  STX     TEMPI3
FF0F   36               FSHA
FF10   8608             LCAA    ¬$$08
FF12   BDF9DE   DLOOP1  JSR     DELAY
FF15   4A               CECA
FF16   26FA             ENE     DLOOP1
FF18   32               FLLA
FF19   DE02             LCX     TEMPI3
FF1B   39               FTS
                *************************
                *↑THIS SLEROUTINE DUMPS ALL OF THE
                *READINGS THAT HAVE BEEN TAKEN TO
                *THE FERIFHERAL DEVICE (IF SWITCHED
                *ON) AND TO THE TERMINAL. ↑THE
                *READINGS ARE PRECEDED BY SOME
                *INITIALIZATION  PARAMTERS (SEE
                *THE OPERATION MANUAL FOR DETAILS).
FF1C   B6E002   DUMP    LCAA    CLOCKB
FF1F   8408             ANDA    ¬$$08              ;CHECK IF DUMP SWITCH PRESSED
FF21   26F9             ENE     DUMP               ;LOOP UNTIL IT IS PRESSED
FF23   BDF629           JSR     ENMEM
FF26   8680             LCAA    ¬$$80
FF28   BA4000           CFAA    STATUS
FF2B   B74000           STAA    STATUS
FF2E   BDFCE6           JSR     SPRINT             ;PRINT THE STARTING DATE AND TIME
FF31   BDFCFF           JSR     FPRINT             ;PRINT THE STOPPING DATE AND TIME
FF34   BDFD16           JSR     WPRINT             ;PRINT THE WARMUP TIME
FF37   BDF555           JSR     LFCR               ;SKIP A LINE
FF3A   BDFD22           JSR     FRPRINT            ;PRINT THE FREQ. OF MEASUREMENTS
FF3D   B6402B           LCAA    NPORTS
FF40   BDF57A           JSR     BYTOUT             ;PRINT THE ¬$ OF READINGS
                                                   ;TAKEN EACH TIME
FF43   CE4040           LCX     ¬$STARTM           ;↑I↑R=STARTING LOCATION OF DATA
FF46   BDF555           JSR     LFCR
FF49   B6402A           LCAA    PORTBT
FF4C   BDF57A           JSR     BYTOUT
FF4F   BDF555   OUTLOP  JSR     LFCR               ;SKIP A LINE
FF52   BC4025           CFX     MDATAH             ;IS IT THE END OF THE READINGS
FF55   2747             EEQ     HELLO              ;IF YES THEN GOTO HELLO
FF57   F6402B           LCAB    NPORTS             ;↑B=¬$ OF READINGS
FF5A   A600             LCAA    $00,X
FF5C   BDF57A           JSR     BYTOUT             ;PRINT TEMP READING
FF5F   08               INX                        ;INCREMENT ↑I↑R
FF60   BDF600           JSR     PSP                ;SKIP A SPACE
FF63   A600             LCAA    $00,X
FF65   BDF57A           JSR     BYTOUT             ;PRINT DEW POINT
FF68   08               INX                        ;INCREMENT ↑I↑R
FF69   5A               CECB
FF6A   5A               CECB                       ;DECREMENT ↑B TWICE
FF6B   BDF600   FULBIT  JSR     PSP                ;SKIP A SPACE
FF6E   A600             LCAA    $00,X
FF70   BDF57A           JSR     BYTOUT             ;PRINT FIRST BYTE
FF73   08               INX                        ;INCREMENT ↑I↑R
FF74   A600             LCAA    $00,X
FF76   44               LSRA
FF77   44               LSRA
FF78   44               LSRA
```

```
FF79   44                LSRA                      ;DROP 8 LOW ORDER BITS
FF7A   BDF68D            JSR     HEXASC            ;CONVERT TO HEX
FF7D   BDF2A0            JSR     OUTA              ;PRINT
FF80   08                INX                       ;INCREMENT ↑I↑R
FF81   5A                DECB                      ;DECREMENT ¬$ OF READINGS PRINTED
FF82   27CB              BEQ     OUTLOP            ;IF DONE BRANCH BACK TO OUTLOP
FF84   BDF60D            JSR     PSP               ;SKIP A SPACE
FF87   09                DEX                       ;DECREMENT ↑I↑R
FF88   A600              LDAA    $00,X
FF8A   840F              ANDA    ¬$$0F              ;MASK OUT HIGH 8 BITS
FF8C   BDF66D            JSR     HEXASC            ;CHANGE TO HEX
FF8F   BDF2A0            JSR     OUTA              ;PRINT THE CHARACTER
FF92   08                INX                       ;INCREMENT ↑I↑R
FF93   A600              LDAA    $00,X
FF95   BDF57A            JSR     BYTOUT            ;PRINT THE BYTE
FF98   08                INX                       ;INCREMENT ↑I↑R
FF99   5A                DECB                      ;DECREMENT ¬$ OF READINGS PRINTED
FF9A   27B3              BEQ     OUTLOP            ;BRANCH BACK IF ROW IS FINISHED
FF9C   20CD              BRA     FULBIT            ;FINISH THE ROW
FF9E   867F      HELLO   LDAA    ¬$$7F
FFA0   B44000            ANDA    STATUS
FFA3   B74000            STAA    STATUS
FFA6   BDF4A3            JSR     DISMEM
FFA9   7EFF1C            JMP     DUMP
FFAC   444154    HEAD1   FCC     ↑7DATE  TIME↑7
FFAF   452020
FFB2   54494D
FFB5   45
FFB6   00                FCB     00
FFB7   202054    HEAD2   FCC     ↑7  TEMP  DEW PT  PRESS ↑7
FFBA   454D50
FFBD   202044
FFC0   455720
FFC3   505420
FFC6   205052
FFC9   455353
FFCC   20
FFCD   00                FCB     00
FFCE   20504F    PORT    FCC     ↑7 PORT↑7
FFD1   5254
FFD3   00                FCB     00
                         END
```

* THE BINARY IN IN PHYSICAL BLOCK 2

------ SYMBOL LEGEND -------

↑	SHIFTED KEY	∧	SUPERSCRIPT
∨	SUBSCRIPT	≤	BACKSPACE
↓	CARRIAGE RETURN	≥	FONT
¬	ACCESS	≠	TIMES
≡	DIVIDE	↱	ASSIGN
%	PERCENT		

CUMULATIVE LIST OF RADIO RESEARCH LABORATORY REPORTS PREPARED UNDER NASA GRANT NSG-5049

RRL Rep. No. 469 - Gardner, C. S. (December 1975), The Effects of Random Path Fluctuations on the Accuracy of Laser Ranging Systems.

RRL Rep. No. 471 - Zanter, D. L., C. S. Gardner and N. N. Rao (January 1976), The Effects of Atmospheric Refraction on the Accuracy of Laser Ranging Systems.

RRL Rep. No. 477 - Gardner, C. S. and J. R. Rowlett (November 1976), Atmospheric Refraction Errors in Laser Ranging Data.

RRL Rep. No. 478 - Gardner, C. S. and B. E. Hendrickson (December 1976), Correction of Laser Ranging Data for the Effects of Horizontal Refractivity Gradients.

RRL Rep. No. 481 - Gardner, C. S. (January 1977), Statistics of the Residual Refraction Errors in Laser Ranging Data

RRL Rep. No. 486 - Gardner, C. S. (June 1977), Comparison Between the Refraction Error Covariance Model and Ray Tracing.

RRL Rep. No. 488 - Gardner, C. S. (December 1977), Speckle Noise in Satellite Based Lidar Systems.

RRL Rep. No. 495 - Gardner, C. S. and G. S. Mecherle (April 1978), Speckle Noise in Direct-Detection Lidar Systems.

RRL Rep. No. 496 - Gardner, C. S. and A. M. Saleh (October 1978), Speckle Noise in Differential Absorption Lidar Systems.

RRL Rep. No. 499 - Gardner, C. S. (January 1979), A Technique for Remotely Measuring surface Pressure from a Satellite Using a Multicolor Laser Ranging System.

PAPERS PUBLISHED

C. S. Gardner, "Effects of Random Path Fluctuations on the Accuracy of Laser Ranging Data," Applied Optics, 15, 2539-2545, October 1976.

C. S. Gardner, "Effects of Horizontal Refractivity Gradients on the Accuracy of Laser Ranging to Satellites," Radio Science, 11, 1037-1044, December 1976.

C. S. Gardner, "Correction of Laser Tracking Data for the Effects of Horizontal Refractivity Gradients," Applied Optics, 16, 2427-2432, September 1977.

C. S. Gardner, J. R. Rowlett, and B. E. Hendrickson, "Ray Tracing Evaluation of a Technique for Correcting the Refraction Errors in Satellite Tracking Data," Applied Optics, 17, 3143-3145, October 1978.